

MAGRATHEA/PATHFINDER

Generated by Doxygen 1.8.2

Wed Oct 6 2021 17:41:42

Contents

1	Namespace Index	1
1.1	Namespace List	1
2	Hierarchical Index	3
2.1	Class Hierarchy	3
3	Class Index	5
3.1	Class List	5
4	File Index	9
4.1	File List	9
5	Namespace Documentation	13
5.1	magrathea Namespace Reference	13
5.1.1	Function Documentation	18
5.1.1.1	operator!=	18
5.1.1.2	operator%	19
5.1.1.3	operator&	19
5.1.1.4	operator&&	20
5.1.1.5	operator*	20
5.1.1.6	operator+	21
5.1.1.7	operator-	21
5.1.1.8	operator/	22
5.1.1.9	operator<	22
5.1.1.10	operator<<	23
5.1.1.11	operator<<	23
5.1.1.12	operator<<	23
5.1.1.13	operator<<	24
5.1.1.14	operator<<	24
5.1.1.15	operator<<	25
5.1.1.16	operator<<	25
5.1.1.17	operator<<	25
5.1.1.18	operator<<	26

5.1.1.19	operator<<	26
5.1.1.20	operator<<	26
5.1.1.21	operator<<	27
5.1.1.22	operator<<	27
5.1.1.23	operator<<	28
5.1.1.24	operator<<	28
5.1.1.25	operator<=	29
5.1.1.26	operator==	29
5.1.1.27	operator>	30
5.1.1.28	operator>=	30
5.1.1.29	operator>>	31
5.1.1.30	operator>>	31
5.1.1.31	operator^	32
5.1.1.32	operator	32
5.1.1.33	operator	33
6	Class Documentation	35
6.1	magrathea::AboutInstitute Exception Reference	35
6.1.1	Detailed Description	37
6.1.2	Constructor & Destructor Documentation	37
6.1.2.1	AboutInstitute	37
6.1.3	Member Function Documentation	37
6.1.3.1	address	37
6.1.3.2	address	38
6.1.3.3	cea	38
6.1.3.4	city	38
6.1.3.5	city	38
6.1.3.6	cnes	39
6.1.3.7	cnrs	39
6.1.3.8	contact	39
6.1.3.9	contact	39
6.1.3.10	country	39
6.1.3.11	country	40
6.1.3.12	esa	40
6.1.3.13	example	40
6.1.3.14	extended	40
6.1.3.15	extended	40
6.1.3.16	iap	41
6.1.3.17	ias	41
6.1.3.18	identification	41

6.1.3.19	identification	41
6.1.3.20	ipag	42
6.1.3.21	link	42
6.1.3.22	link	42
6.1.3.23	luth	42
6.1.3.24	name	42
6.1.3.25	name	43
6.1.3.26	obspm	43
6.1.3.27	region	43
6.1.3.28	region	43
6.1.3.29	sap	43
6.1.3.30	street	44
6.1.3.31	street	44
6.1.3.32	title	44
6.1.3.33	title	44
6.1.3.34	zip	44
6.1.3.35	zip	45
6.1.4	Member Data Documentation	45
6.1.4.1	operator	45
6.2	magrathea::AboutLicense Exception Reference	45
6.2.1	Detailed Description	47
6.2.2	Constructor & Destructor Documentation	47
6.2.2.1	AboutLicense	47
6.2.3	Member Function Documentation	47
6.2.3.1	bsd	47
6.2.3.2	cc0v1	48
6.2.3.3	ccbyncndv3	48
6.2.3.4	ccbyncsav3	48
6.2.3.5	ccbyncv3	48
6.2.3.6	ccbyndv3	48
6.2.3.7	ccbysav3	48
6.2.3.8	ccbyv3	49
6.2.3.9	cecill	49
6.2.3.10	cecillb	49
6.2.3.11	cecillc	49
6.2.3.12	description	49
6.2.3.13	description	49
6.2.3.14	example	50
6.2.3.15	freebsd	50
6.2.3.16	gplv3	50

6.2.3.17	lgplv3	50
6.2.3.18	link	50
6.2.3.19	link	51
6.2.3.20	modifiedbsd	51
6.2.3.21	name	51
6.2.3.22	name	51
6.2.3.23	proprietary	51
6.2.3.24	proprietary	52
6.2.3.25	reference	52
6.2.3.26	reference	52
6.2.3.27	text	52
6.2.3.28	text	52
6.2.3.29	text	53
6.2.3.30	title	53
6.2.3.31	title	53
6.2.3.32	version	53
6.2.3.33	version	53
6.2.4	Member Data Documentation	54
6.2.4.1	operator	54
6.3	magrathea::AboutObject< Types > Exception Template Reference	54
6.3.1	Detailed Description	55
6.3.2	Constructor & Destructor Documentation	55
6.3.2.1	AboutObject	55
6.3.3	Member Function Documentation	55
6.3.3.1	example	55
6.3.4	Member Data Documentation	55
6.3.4.1	operator	55
6.4	magrathea::AboutPeople Exception Reference	55
6.4.1	Detailed Description	57
6.4.2	Constructor & Destructor Documentation	57
6.4.2.1	AboutPeople	57
6.4.3	Member Function Documentation	58
6.4.3.1	altnail	58
6.4.3.2	altnail	58
6.4.3.3	begin	58
6.4.3.4	begin	58
6.4.3.5	contact	58
6.4.3.6	contact	59
6.4.3.7	end	59
6.4.3.8	end	59

6.4.3.9	example	59
6.4.3.10	first	59
6.4.3.11	first	60
6.4.3.12	last	60
6.4.3.13	last	60
6.4.3.14	link	60
6.4.3.15	link	60
6.4.3.16	mail	61
6.4.3.17	mail	61
6.4.3.18	mails	61
6.4.3.19	mails	61
6.4.3.20	name	62
6.4.3.21	name	62
6.4.3.22	vreverdy	62
6.4.3.23	years	62
6.4.3.24	years	63
6.4.4	Member Data Documentation	63
6.4.4.1	operator	63
6.5	magrathea::AbstractAboutObject< Crtp, Types > Class Template Reference	63
6.5.1	Detailed Description	65
6.5.2	Constructor & Destructor Documentation	65
6.5.2.1	~AbstractAboutObject	65
6.5.2.2	AbstractAboutObject	65
6.5.2.3	AbstractAboutObject	65
6.5.2.4	AbstractAboutObject	66
6.5.2.5	AbstractAboutObject	66
6.5.3	Member Function Documentation	66
6.5.3.1	assign	66
6.5.3.2	assign	66
6.5.3.3	assign	67
6.5.3.4	assign	67
6.5.3.5	assign	67
6.5.3.6	cast	68
6.5.3.7	copy	68
6.5.3.8	data	68
6.5.3.9	data	69
6.5.3.10	data	69
6.5.3.11	nullify	70
6.5.3.12	operator!=	70
6.5.3.13	operator=	71

6.5.3.14	<code>operator=</code>	71
6.5.3.15	<code>operator=</code>	71
6.5.3.16	<code>operator==</code>	72
6.6	<code>magrathea::AbstractContents< Crtp, Category, Types ></code> Class Template Reference	72
6.6.1	Detailed Description	74
6.6.2	Constructor & Destructor Documentation	74
6.6.2.1	<code>~AbstractContents</code>	74
6.6.2.2	<code>AbstractContents</code>	74
6.6.2.3	<code>AbstractContents</code>	74
6.6.2.4	<code>AbstractContents</code>	75
6.6.2.5	<code>AbstractContents</code>	75
6.6.3	Member Function Documentation	75
6.6.3.1	<code>assign</code>	75
6.6.3.2	<code>assign</code>	76
6.6.3.3	<code>assign</code>	76
6.6.3.4	<code>assign</code>	76
6.6.3.5	<code>assign</code>	77
6.6.3.6	<code>cast</code>	77
6.6.3.7	<code>copy</code>	77
6.6.3.8	<code>data</code>	77
6.6.3.9	<code>data</code>	78
6.6.3.10	<code>data</code>	78
6.6.3.11	<code>nullify</code>	79
6.6.3.12	<code>operator!=</code>	79
6.6.3.13	<code>operator=</code>	80
6.6.3.14	<code>operator=</code>	80
6.6.3.15	<code>operator=</code>	81
6.6.3.16	<code>operator==</code>	81
6.7	<code>magrathea::AbstractHyperCube< Crtp, Dimension, Vector, Scalar ></code> Class Template Reference	81
6.7.1	Detailed Description	83
6.7.2	Constructor & Destructor Documentation	84
6.7.2.1	<code>~AbstractHyperCube</code>	84
6.7.3	Member Function Documentation	84
6.7.3.1	<code>area</code>	84
6.7.3.2	<code>center</code>	84
6.7.3.3	<code>center</code>	84
6.7.3.4	<code>diagonal</code>	85
6.7.3.5	<code>dimension</code>	85
6.7.3.6	<code>elements</code>	85
6.7.3.7	<code>example</code>	85

6.7.3.8	inside	86
6.7.3.9	length	86
6.7.3.10	maximum	86
6.7.3.11	maximum	86
6.7.3.12	minimum	87
6.7.3.13	minimum	87
6.7.3.14	outside	87
6.7.3.15	random	87
6.7.3.16	random	88
6.7.3.17	subelements	88
6.7.3.18	surface	89
6.7.3.19	volume	89
6.8	magrathea::AbstractHyperSphere< Crtp, Dimension, Vector, Scalar > Class Template Reference	89
6.8.1	Detailed Description	91
6.8.2	Constructor & Destructor Documentation	91
6.8.2.1	~AbstractHyperSphere	91
6.8.3	Member Function Documentation	91
6.8.3.1	center	91
6.8.3.2	center	92
6.8.3.3	diameter	92
6.8.3.4	dimension	92
6.8.3.5	example	92
6.8.3.6	inside	93
6.8.3.7	maximum	93
6.8.3.8	maximum	93
6.8.3.9	minimum	93
6.8.3.10	minimum	94
6.8.3.11	outside	94
6.8.3.12	radius	94
6.8.3.13	random	94
6.8.3.14	random	95
6.8.3.15	sn	95
6.8.3.16	surface	96
6.8.3.17	uniform	96
6.8.3.18	volume	96
6.9	magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters > Class Template Reference	97
6.9.1	Detailed Description	102
6.9.2	Constructor & Destructor Documentation	102
6.9.2.1	~AbstractNArray	102
6.9.2.2	AbstractNArray	102

6.9.2.3	AbstractNArray	103
6.9.2.4	AbstractNArray	103
6.9.2.5	AbstractNArray	103
6.9.3	Member Function Documentation	103
6.9.3.1	aarrange	104
6.9.3.2	abs	104
6.9.3.3	amax	104
6.9.3.4	amin	105
6.9.3.5	apply	105
6.9.3.6	apply	106
6.9.3.7	arrange	106
6.9.3.8	asort	107
6.9.3.9	begin	107
6.9.3.10	begin	108
6.9.3.11	cbegin	108
6.9.3.12	cend	108
6.9.3.13	crbegin	108
6.9.3.14	crend	109
6.9.3.15	data	109
6.9.3.16	data	109
6.9.3.17	distinct	109
6.9.3.18	end	110
6.9.3.19	end	110
6.9.3.20	eq	110
6.9.3.21	example	110
6.9.3.22	index	111
6.9.3.23	indexed	111
6.9.3.24	log	112
6.9.3.25	log	112
6.9.3.26	mean	112
6.9.3.27	modify	113
6.9.3.28	modify	113
6.9.3.29	ne	114
6.9.3.30	norm	114
6.9.3.31	normalize	115
6.9.3.32	null	115
6.9.3.33	one	116
6.9.3.34	operator[]	116
6.9.3.35	operator[]	116
6.9.3.36	pow	116

6.9.3.37	pow	117
6.9.3.38	progressive	117
6.9.3.39	random	118
6.9.3.40	random	118
6.9.3.41	rbegin	119
6.9.3.42	rbegin	119
6.9.3.43	rearrange	119
6.9.3.44	rend	119
6.9.3.45	rend	120
6.9.3.46	renormalize	120
6.9.3.47	resort	120
6.9.3.48	rt	121
6.9.3.49	rt	121
6.9.3.50	sigma	122
6.9.3.51	sort	122
6.9.3.52	unity	123
6.9.3.53	value	123
6.9.3.54	zero	123
6.9.4	Member Data Documentation	124
6.9.4.1	_data	124
6.9.4.2	operator	124
6.10	magrathea::AbstractShape Class Reference	124
6.10.1	Detailed Description	125
6.10.2	Constructor & Destructor Documentation	125
6.10.2.1	~AbstractShape	125
6.10.2.2	AbstractShape	125
6.10.3	Member Function Documentation	126
6.10.3.1	alt	126
6.10.3.2	binomial	126
6.10.3.3	combinations	126
6.10.3.4	example	127
6.10.3.5	factorial	127
6.10.3.6	factorial	127
6.10.3.7	golden	127
6.10.3.8	pi	128
6.10.3.9	pow	128
6.10.3.10	rt	128
6.10.3.11	sqrtpi	129
6.10.3.12	variations	129
6.11	magrathea::AbstractStep< Crtp, Scalar, Array, Tuple > Class Template Reference	129

6.11.1	Detailed Description	131
6.11.2	Constructor & Destructor Documentation	131
6.11.2.1	<code>~AbstractStep</code>	131
6.11.2.2	<code>AbstractStep</code>	131
6.11.2.3	<code>AbstractStep</code>	132
6.11.2.4	<code>AbstractStep</code>	132
6.11.3	Member Function Documentation	132
6.11.3.1	<code>assign</code>	132
6.11.3.2	<code>assign</code>	132
6.11.3.3	<code>assign</code>	133
6.11.3.4	<code>assign</code>	133
6.11.3.5	<code>cast</code>	134
6.11.3.6	<code>copy</code>	134
6.11.3.7	<code>data</code>	134
6.11.3.8	<code>data</code>	135
6.11.3.9	<code>data</code>	135
6.11.3.10	<code>nullify</code>	136
6.11.3.11	<code>operator!=</code>	136
6.11.3.12	<code>operator=</code>	137
6.11.3.13	<code>operator=</code>	137
6.11.3.14	<code>operator==</code>	137
6.12	<code>magrathea::AbstractSubstance< Crtp, Types ></code> Class Template Reference	138
6.12.1	Detailed Description	139
6.12.2	Constructor & Destructor Documentation	140
6.12.2.1	<code>~AbstractSubstance</code>	140
6.12.2.2	<code>AbstractSubstance</code>	140
6.12.2.3	<code>AbstractSubstance</code>	140
6.12.2.4	<code>AbstractSubstance</code>	140
6.12.2.5	<code>AbstractSubstance</code>	141
6.12.3	Member Function Documentation	141
6.12.3.1	<code>assign</code>	141
6.12.3.2	<code>assign</code>	141
6.12.3.3	<code>assign</code>	141
6.12.3.4	<code>assign</code>	142
6.12.3.5	<code>assign</code>	142
6.12.3.6	<code>cast</code>	142
6.12.3.7	<code>copy</code>	143
6.12.3.8	<code>data</code>	143
6.12.3.9	<code>data</code>	144
6.12.3.10	<code>data</code>	144

6.12.3.11	nullify	145
6.12.3.12	operator!=	145
6.12.3.13	operator=	145
6.12.3.14	operator=	146
6.12.3.15	operator=	146
6.12.3.16	operator==	146
6.13	Catalogues Class Reference	147
6.13.1	Member Function Documentation	148
6.13.1.1	iterateNewtonMethod	148
6.13.1.2	newtonMethod2d	148
6.13.1.3	ReadParamFile	149
6.13.1.4	ReadParticlesASCII	150
6.13.1.5	ReadParticlesHDF5	150
6.13.1.6	relCat	150
6.13.1.7	relCat_with_previous_cat	151
6.14	Cone< Vector, Scalar > Exception Template Reference	151
6.14.1	Detailed Description	153
6.14.2	Constructor & Destructor Documentation	153
6.14.2.1	Cone	153
6.14.3	Member Function Documentation	154
6.14.3.1	angle	154
6.14.3.2	angle	154
6.14.3.3	base	155
6.14.3.4	base	155
6.14.3.5	circle	155
6.14.3.6	diameter	155
6.14.3.7	direction	156
6.14.3.8	direction	156
6.14.3.9	example	156
6.14.3.10	inside	156
6.14.3.11	length	157
6.14.3.12	outside	157
6.14.3.13	radius	157
6.14.3.14	surface	157
6.14.3.15	vertex	158
6.14.3.16	vertex	158
6.14.3.17	volume	158
6.14.4	Member Data Documentation	158
6.14.4.1	operator	159
6.15	magrathea::Constant< Type, Size > Exception Template Reference	159

6.15.1 Detailed Description	164
6.15.2 Constructor & Destructor Documentation	164
6.15.2.1 Constant	164
6.15.2.2 Constant	164
6.15.2.3 Constant	165
6.15.2.4 Constant	165
6.15.2.5 Constant	165
6.15.2.6 Constant	166
6.15.3 Member Function Documentation	166
6.15.3.1 abs	166
6.15.3.2 abs	166
6.15.3.3 add	167
6.15.3.4 add	167
6.15.3.5 cast	167
6.15.3.6 cb	168
6.15.3.7 cb	168
6.15.3.8 copy	168
6.15.3.9 data	169
6.15.3.10 div	169
6.15.3.11 div	169
6.15.3.12 example	170
6.15.3.13 get	170
6.15.3.14 icbrt	170
6.15.3.15 icbrt	170
6.15.3.16 ilog	171
6.15.3.17 ilog	171
6.15.3.18 ilog10	172
6.15.3.19 ilog10	172
6.15.3.20 ilog2	172
6.15.3.21 ilog2	173
6.15.3.22 inv	173
6.15.3.23 inv	173
6.15.3.24 irt	174
6.15.3.25 irt	174
6.15.3.26 isqrt	174
6.15.3.27 isqrt	175
6.15.3.28 metailog	175
6.15.3.29 metairt	176
6.15.3.30 metapow	176
6.15.3.31 mod	177

6.15.3.32 mod	177
6.15.3.33 mul	177
6.15.3.34 mul	178
6.15.3.35 nullify	178
6.15.3.36 operator Type	178
6.15.3.37 operator()	178
6.15.3.38 operator()	179
6.15.3.39 operator()	179
6.15.3.40 operator[]	179
6.15.3.41 opp	180
6.15.3.42 opp	180
6.15.3.43 pow	180
6.15.3.44 pow	181
6.15.3.45 ratio	181
6.15.3.46 ratio	182
6.15.3.47 resize	182
6.15.3.48 set	182
6.15.3.49 set	183
6.15.3.50 sgn	183
6.15.3.51 sgn	183
6.15.3.52 si	184
6.15.3.53 si	184
6.15.3.54 size	184
6.15.3.55 sq	184
6.15.3.56 sq	185
6.15.3.57 sub	185
6.15.3.58 sub	185
6.15.3.59 transmute	186
6.15.3.60 transmute	186
6.15.3.61 value	187
6.15.4 Friends And Related Function Documentation	187
6.15.4.1 operator<<	187
6.15.5 Member Data Documentation	187
6.15.5.1 __data	187
6.16 magrathea::Constants< Type > Exception Template Reference	188
6.16.1 Detailed Description	195
6.16.2 Member Function Documentation	195
6.16.2.1 alpha	195
6.16.2.2 alpha2	196
6.16.2.3 arcmin	196

6.16.2.4	arcmin2	196
6.16.2.5	arcsec	196
6.16.2.6	arcsec2	196
6.16.2.7	atm	197
6.16.2.8	atomic	197
6.16.2.9	au	197
6.16.2.10	c	197
6.16.2.11	c2	197
6.16.2.12	c4	198
6.16.2.13	catalan	198
6.16.2.14	celsius	198
6.16.2.15	day	198
6.16.2.16	deg	198
6.16.2.17	deg2	199
6.16.2.18	e	199
6.16.2.19	e2	199
6.16.2.20	epsilon0	199
6.16.2.21	euler	199
6.16.2.22	ev	200
6.16.2.23	evc2	200
6.16.2.24	example	200
6.16.2.25	faraday	200
6.16.2.26	fcd	200
6.16.2.27	feigenbauma	201
6.16.2.28	feigenbaumd	201
6.16.2.29	g	201
6.16.2.30	g2	201
6.16.2.31	gas	201
6.16.2.32	glaisher	202
6.16.2.33	gn	202
6.16.2.34	golden	202
6.16.2.35	h	202
6.16.2.36	h2	202
6.16.2.37	hartree	203
6.16.2.38	hbar	203
6.16.2.39	hbar2	203
6.16.2.40	hfs	203
6.16.2.41	hour	203
6.16.2.42	josephson	204
6.16.2.43	kb	204

6.16.2.44	kb2	204
6.16.2.45	kcd	204
6.16.2.46	ke	204
6.16.2.47	khinchin	205
6.16.2.48	klitzing	205
6.16.2.49	ly	205
6.16.2.50	magnetob	205
6.16.2.51	magnetonn	205
6.16.2.52	mbosonw	206
6.16.2.53	mbosonz	206
6.16.2.54	mearth	206
6.16.2.55	melectron	206
6.16.2.56	mertens	206
6.16.2.57	mgluon	207
6.16.2.58	mgraviton	207
6.16.2.59	mhiggs	207
6.16.2.60	minute	207
6.16.2.61	mjupiter	207
6.16.2.62	mmars	208
6.16.2.63	mm mercury	208
6.16.2.64	mmoon	208
6.16.2.65	mmuon	208
6.16.2.66	mneptune	208
6.16.2.67	mneutron	209
6.16.2.68	mnuelectron	209
6.16.2.69	mnumuon	209
6.16.2.70	mnutau	209
6.16.2.71	mphoton	209
6.16.2.72	mproton	210
6.16.2.73	mquarkb	210
6.16.2.74	mquarkc	210
6.16.2.75	mquarkd	210
6.16.2.76	mquarks	210
6.16.2.77	mquarkt	211
6.16.2.78	mquarku	211
6.16.2.79	msaturn	211
6.16.2.80	msun	211
6.16.2.81	mtau	211
6.16.2.82	mu0	212
6.16.2.83	muranus	212

6.16.2.84 mvenus	212
6.16.2.85 na	212
6.16.2.86 napier	212
6.16.2.87 omega	213
6.16.2.88 pc	213
6.16.2.89 pi	213
6.16.2.90 pi2	213
6.16.2.91 plancke	213
6.16.2.92 planckf	214
6.16.2.93 planckl	214
6.16.2.94 planckm	214
6.16.2.95 planckp	214
6.16.2.96 planckq	214
6.16.2.97 planckt	215
6.16.2.98 plancktheta	215
6.16.2.99 plastic	215
6.16.2.100pythagoras	215
6.16.2.101quantumc	215
6.16.2.102quantumf	216
6.16.2.103radiationf	216
6.16.2.104radiations	216
6.16.2.105bohr	216
6.16.2.106earth	216
6.16.2.107electron	217
6.16.2.108jupiter	217
6.16.2.109mars	217
6.16.2.110mercury	217
6.16.2.111rmoon	217
6.16.2.112rneptune	218
6.16.2.113rsaturn	218
6.16.2.114rsun	218
6.16.2.115ruranus	218
6.16.2.116venus	218
6.16.2.117rydberg	219
6.16.2.118second	219
6.16.2.119soldner	219
6.16.2.120sphere	219
6.16.2.121sqrtpi	219
6.16.2.122stefan	220
6.16.2.123tanarcsec	220

6.16.2.12 <code>theodorus</code>	220
6.16.2.12 <code>thomson</code>	220
6.16.2.12 <code>wien</code>	220
6.16.2.12 <code>year</code>	221
6.16.2.12 <code>z0</code>	221
6.17 <code>magrathea::Contents< Category, Types ></code> Exception Template Reference	221
6.17.1 Detailed Description	222
6.17.2 Constructor & Destructor Documentation	222
6.17.2.1 Contents	222
6.17.3 Member Function Documentation	222
6.17.3.1 example	222
6.17.4 Member Data Documentation	222
6.17.4.1 operator	222
6.18 <code>Create_octree</code> Class Reference	223
6.18.1 Member Function Documentation	224
6.18.1.1 <code>CreateOctreeWithCIC</code>	224
6.18.1.2 <code>CreateOctreeWithTSC</code>	225
6.18.1.3 <code>PreparationASCII</code>	226
6.18.1.4 <code>PreparationASCII</code>	226
6.18.1.5 <code>PreparationBinary</code>	227
6.18.1.6 <code>PreparationBinary</code>	227
6.18.1.7 <code>PreparationHDF5_from_cells</code>	228
6.18.1.8 <code>PreparationHDF5_from_particles</code>	229
6.18.1.9 <code>PreparationHDF5_from_particles</code>	229
6.18.1.10 <code>ReadParamFile</code>	230
6.19 <code>magrathea::DataHandler</code> Exception Reference	230
6.19.1 Detailed Description	237
6.19.2 Member Function Documentation	237
6.19.2.1 <code>array</code>	237
6.19.2.2 <code>array</code>	238
6.19.2.3 <code>bytesize</code>	238
6.19.2.4 <code>bytesize</code>	238
6.19.2.5 <code>bytesize</code>	238
6.19.2.6 <code>bytesize</code>	238
6.19.2.7 <code>bytesize</code>	239
6.19.2.8 <code>bytesize</code>	239
6.19.2.9 <code>bytesize</code>	240
6.19.2.10 <code>byteswap</code>	240
6.19.2.11 <code>byteswap</code>	240
6.19.2.12 <code>byteswap</code>	241

6.19.2.13 byteswap	241
6.19.2.14 byteswap	241
6.19.2.15 byteswap	242
6.19.2.16 byteswap	242
6.19.2.17 equalize	242
6.19.2.18 equalize	243
6.19.2.19 equalize	243
6.19.2.20 equalize	243
6.19.2.21 equalize	244
6.19.2.22 equalize	244
6.19.2.23 example	245
6.19.2.24 hexify	245
6.19.2.25 hexify	245
6.19.2.26 hexify	245
6.19.2.27 hexify	246
6.19.2.28 hexify	246
6.19.2.29 hexify	247
6.19.2.30 nullify	247
6.19.2.31 nullify	247
6.19.2.32 nullify	247
6.19.2.33 nullify	248
6.19.2.34 nullify	248
6.19.2.35 nullify	248
6.19.2.36 print	249
6.19.2.37 print	249
6.19.2.38 print	249
6.19.2.39 print	250
6.19.2.40 print	250
6.19.2.41 print	250
6.19.2.42 rbytesize	251
6.19.2.43 rbytesize	251
6.19.2.44 rbytesize	251
6.19.2.45 rbyteswap	252
6.19.2.46 rbyteswap	252
6.19.2.47 rbyteswap	252
6.19.2.48 rbyteswap	253
6.19.2.49 read	253
6.19.2.50 read	253
6.19.2.51 read	254
6.19.2.52 read	254

6.19.2.53 read	254
6.19.2.54 read	255
6.19.2.55 read	255
6.19.2.56 read	256
6.19.2.57 read	256
6.19.2.58 read	256
6.19.2.59 read	257
6.19.2.60 read	257
6.19.2.61 realuze	258
6.19.2.62 realuze	258
6.19.2.63 realuze	258
6.19.2.64 rhexify	259
6.19.2.65 rhexify	259
6.19.2.66 rhexify	260
6.19.2.67 rnullify	260
6.19.2.68 rnullify	260
6.19.2.69 rnullify	261
6.19.2.70 rprint	261
6.19.2.71 rprint	261
6.19.2.72 rprint	262
6.19.2.73 rread	262
6.19.2.74 rread	262
6.19.2.75 rread	263
6.19.2.76 rread	263
6.19.2.77 rread	263
6.19.2.78 rread	264
6.19.2.79 rscan	264
6.19.2.80 rscan	265
6.19.2.81 rscan	265
6.19.2.82 rsize	265
6.19.2.83 rsize	266
6.19.2.84 rsize	266
6.19.2.85 rstringify	266
6.19.2.86 rstringify	267
6.19.2.87 rstringify	267
6.19.2.88 rwrite	268
6.19.2.89 rwrite	268
6.19.2.90 rwrite	268
6.19.2.91 rwrite	269
6.19.2.92 rwrite	269

6.19.2.93 <code>rwrite</code>	269
6.19.2.94 <code>scan</code>	270
6.19.2.95 <code>scan</code>	270
6.19.2.96 <code>scan</code>	270
6.19.2.97 <code>scan</code>	271
6.19.2.98 <code>scan</code>	271
6.19.2.99 <code>scan</code>	271
6.19.2.100 <code>size</code>	272
6.19.2.101 <code>size</code>	272
6.19.2.102 <code>size</code>	272
6.19.2.103 <code>size</code>	273
6.19.2.104 <code>size</code>	273
6.19.2.105 <code>size</code>	273
6.19.2.106 <code>stringify</code>	274
6.19.2.107 <code>stringify</code>	274
6.19.2.108 <code>stringify</code>	274
6.19.2.109 <code>stringify</code>	275
6.19.2.110 <code>stringify</code>	275
6.19.2.111 <code>stringify</code>	276
6.19.2.112 <code>tuple</code>	276
6.19.2.113 <code>tuple</code>	276
6.19.2.114 <code>write</code>	276
6.19.2.115 <code>write</code>	277
6.19.2.116 <code>write</code>	277
6.19.2.117 <code>write</code>	277
6.19.2.118 <code>write</code>	278
6.19.2.119 <code>write</code>	278
6.19.2.120 <code>write</code>	279
6.19.2.121 <code>write</code>	279
6.19.2.122 <code>write</code>	279
6.19.2.123 <code>write</code>	280
6.19.2.124 <code>write</code>	280
6.19.2.125 <code>write</code>	281
6.20 <code>magrathea::DataModel</code> Exception Reference	281
6.20.1 Detailed Description	283
6.20.2 Constructor & Destructor Documentation	284
6.20.2.1 <code>DataModel</code>	284
6.20.2.2 <code>DataModel</code>	284
6.20.3 Member Function Documentation	285
6.20.3.1 <code>assign</code>	285

6.20.3.2 assign	285
6.20.3.3 assign	285
6.20.3.4 cast	286
6.20.3.5 check	286
6.20.3.6 clear	287
6.20.3.7 complement	287
6.20.3.8 complement	287
6.20.3.9 control	287
6.20.3.10 control754	288
6.20.3.11 copy	288
6.20.3.12 data	288
6.20.3.13 endianness	288
6.20.3.14 endianness	289
6.20.3.15 example	289
6.20.3.16 fundamental	289
6.20.3.17 get	289
6.20.3.18 ieee754	290
6.20.3.19 ieee754	290
6.20.3.20 operator!=	290
6.20.3.21 operator()	290
6.20.3.22 operator==	291
6.20.3.23 set	291
6.20.3.24 size	291
6.20.3.25 size	292
6.20.3.26 size	292
6.20.3.27 system	292
6.20.4 Friends And Related Function Documentation	292
6.20.4.1 operator<<	292
6.20.5 Member Data Documentation	293
6.20.5.1 __code	293
6.21 magrathea::DataSize Exception Reference	293
6.21.1 Detailed Description	296
6.21.2 Constructor & Destructor Documentation	296
6.21.2.1 DataSize	296
6.21.2.2 DataSize	297
6.21.3 Member Function Documentation	297
6.21.3.1 bit	297
6.21.3.2 bits	297
6.21.3.3 bits	297
6.21.3.4 byte	298

6.21.3.5 bytes	298
6.21.3.6 bytes	298
6.21.3.7 cast	299
6.21.3.8 copy	299
6.21.3.9 data	299
6.21.3.10 empty	299
6.21.3.11 empty	299
6.21.3.12 exa	300
6.21.3.13 exabytes	300
6.21.3.14 exabytes	300
6.21.3.15 example	301
6.21.3.16 exbi	301
6.21.3.17 exbibytes	301
6.21.3.18 exbibytes	301
6.21.3.19 gibi	302
6.21.3.20 gibibytes	302
6.21.3.21 gibibytes	302
6.21.3.22 giga	303
6.21.3.23 gigabytes	303
6.21.3.24 gigabytes	303
6.21.3.25 kibi	304
6.21.3.26 kibibytes	304
6.21.3.27 kibibytes	304
6.21.3.28 kilo	304
6.21.3.29 kilobytes	305
6.21.3.30 kilobytes	305
6.21.3.31 mebi	305
6.21.3.32 mebibytes	306
6.21.3.33 mebibytes	306
6.21.3.34 mega	306
6.21.3.35 megabytes	307
6.21.3.36 megabytes	307
6.21.3.37 operator()	307
6.21.3.38 operator=	307
6.21.3.39 operator=	308
6.21.3.40 pebi	308
6.21.3.41 pebibytes	308
6.21.3.42 pebibytes	309
6.21.3.43 peta	309
6.21.3.44 petabytes	309

6.21.3.45 petabytes	310
6.21.3.46 tebi	310
6.21.3.47 tebibytes	310
6.21.3.48 tebibytes	310
6.21.3.49 tera	311
6.21.3.50 terabytes	311
6.21.3.51 terabytes	311
6.21.3.52 valid	312
6.21.3.53 valid	312
6.21.4 Friends And Related Function Documentation	312
6.21.4.1 operator<<	312
6.21.5 Member Data Documentation	312
6.21.5.1 _size	312
6.22 magrathea::EulerianCategory Exception Reference	313
6.22.1 Detailed Description	313
6.22.2 Member Function Documentation	313
6.22.2.1 example	313
6.23 magrathea::Evolution< Type, Container > Exception Template Reference	313
6.23.1 Detailed Description	317
6.23.2 Constructor & Destructor Documentation	317
6.23.2.1 Evolution	317
6.23.2.2 Evolution	317
6.23.2.3 Evolution	317
6.23.3 Member Function Documentation	318
6.23.3.1 append	318
6.23.3.2 assign	318
6.23.3.3 assign	318
6.23.3.4 assign	319
6.23.3.5 assign	319
6.23.3.6 at	319
6.23.3.7 at	320
6.23.3.8 back	320
6.23.3.9 back	320
6.23.3.10 begin	321
6.23.3.11 begin	321
6.23.3.12 capacity	321
6.23.3.13 cast	321
6.23.3.14 cbegin	322
6.23.3.15 cend	322
6.23.3.16 clear	322

6.23.3.17 container	322
6.23.3.18 container	322
6.23.3.19 copy	323
6.23.3.20 crbegin	323
6.23.3.21 crend	323
6.23.3.22 cycle	323
6.23.3.23 cycle	323
6.23.3.24 data	324
6.23.3.25 data	324
6.23.3.26 empty	324
6.23.3.27 end	324
6.23.3.28 end	325
6.23.3.29 example	325
6.23.3.30 front	325
6.23.3.31 front	325
6.23.3.32 fullclear	325
6.23.3.33 index	326
6.23.3.34 nullify	326
6.23.3.35 operator!=	326
6.23.3.36 operator<	327
6.23.3.37 operator<=	327
6.23.3.38 operator=	327
6.23.3.39 operator=	327
6.23.3.40 operator=	328
6.23.3.41 operator==	328
6.23.3.42 operator>	328
6.23.3.43 operator>=	329
6.23.3.44 operator[]	329
6.23.3.45 operator[]	329
6.23.3.46 pop	330
6.23.3.47 rbegin	330
6.23.3.48 rbegin	330
6.23.3.49 rend	330
6.23.3.50 rend	330
6.23.3.51 reserve	331
6.23.3.52 resize	331
6.23.3.53 shrink	331
6.23.3.54 size	331
6.23.3.55 space	332
6.23.4 Friends And Related Function Documentation	332

6.23.4.1 operator<<	332
6.23.5 Member Data Documentation	332
6.23.5.1 _container	332
6.24 magrathea::FileList Exception Reference	332
6.24.1 Detailed Description	337
6.24.2 Constructor & Destructor Documentation	337
6.24.2.1 FileList	337
6.24.2.2 FileList	337
6.24.2.3 FileList	337
6.24.2.4 FileList	338
6.24.2.5 FileList	338
6.24.3 Member Function Documentation	338
6.24.3.1 apply	339
6.24.3.2 assign	339
6.24.3.3 assign	339
6.24.3.4 assign	340
6.24.3.5 assign	340
6.24.3.6 assign	340
6.24.3.7 at	341
6.24.3.8 back	341
6.24.3.9 capacity	341
6.24.3.10 cast	342
6.24.3.11 clear	342
6.24.3.12 cleared	342
6.24.3.13 common	342
6.24.3.14 container	342
6.24.3.15 convert	343
6.24.3.16 copy	343
6.24.3.17 count	343
6.24.3.18 count	343
6.24.3.19 empty	343
6.24.3.20 example	344
6.24.3.21 find	344
6.24.3.22 format	344
6.24.3.23 formatify	344
6.24.3.24 front	345
6.24.3.25 generate	345
6.24.3.26 generate	345
6.24.3.27 generate	346
6.24.3.28 generate	346

6.24.3.29 generator	346
6.24.3.30 get	347
6.24.3.31 length	347
6.24.3.32 numberify	347
6.24.3.33 offset	348
6.24.3.34 operator!=	348
6.24.3.35 operator()	348
6.24.3.36 operator=	348
6.24.3.37 operator=	349
6.24.3.38 operator=	349
6.24.3.39 operator==	349
6.24.3.40 operator[]	350
6.24.3.41 reserve	350
6.24.3.42 resize	350
6.24.3.43 root	350
6.24.3.44 shrink	351
6.24.3.45 size	351
6.24.3.46 sort	351
6.24.3.47 sorted	351
6.24.3.48 unicity	352
6.24.3.49 unique	352
6.24.4 Friends And Related Function Documentation	353
6.24.4.1 operator<<	353
6.24.5 Member Data Documentation	353
6.24.5.1 _container	353
6.24.5.2 _format	353
6.24.5.3 _generator	353
6.24.5.4 _length	353
6.24.5.5 _offset	353
6.24.5.6 _root	353
6.24.5.7 _size	353
6.25 magrathea::FileSystem Exception Reference	354
6.25.1 Detailed Description	356
6.25.2 Member Function Documentation	356
6.25.2.1 ascii	356
6.25.2.2 binary	357
6.25.2.3 bom	357
6.25.2.4 bom	357
6.25.2.5 byteswap	358
6.25.2.6 byteswap	358

6.25.2.7	check	359
6.25.2.8	compare	359
6.25.2.9	compare	359
6.25.2.10	compare	360
6.25.2.11	copy	360
6.25.2.12	create	361
6.25.2.13	dated	361
6.25.2.14	eascii	361
6.25.2.15	empty	362
6.25.2.16	endianness	362
6.25.2.17	exact	362
6.25.2.18	example	363
6.25.2.19	exist	363
6.25.2.20	generate	363
6.25.2.21	generate	364
6.25.2.22	generate	364
6.25.2.23	initialize	365
6.25.2.24	join	365
6.25.2.25	regular	366
6.25.2.26	remove	366
6.25.2.27	rename	366
6.25.2.28	reset	367
6.25.2.29	size	367
6.25.2.30	size	367
6.25.2.31	size	368
6.25.2.32	split	368
6.25.2.33	temporary	368
6.25.2.34	unjoin	369
6.25.2.35	unsplit	369
6.25.2.36	weight	370
6.26	Gravity< Type, Dimension > Exception Template Reference	370
6.26.1	Detailed Description	374
6.26.2	Constructor & Destructor Documentation	375
6.26.2.1	Gravity	375
6.26.2.2	Gravity	375
6.26.3	Member Function Documentation	375
6.26.3.1	a	375
6.26.3.2	a	376
6.26.3.3	a	376
6.26.3.4	a	376

6.26.3.5 dphidt	376
6.26.3.6 dphidt	376
6.26.3.7 dphidt	377
6.26.3.8 dphidt	377
6.26.3.9 dphidx	377
6.26.3.10 dphidx	377
6.26.3.11 dphidx	378
6.26.3.12 dphidx	378
6.26.3.13 dphidxz	378
6.26.3.14 dphidxz	378
6.26.3.15 dphidxz	379
6.26.3.16 dphidxz	379
6.26.3.17 dphidy	379
6.26.3.18 dphidy	380
6.26.3.19 dphidy	380
6.26.3.20 dphidy	380
6.26.3.21 dphidz	380
6.26.3.22 dphidz	381
6.26.3.23 dphidz	381
6.26.3.24 dphidz	381
6.26.3.25 example	381
6.26.3.26 example	381
6.26.3.27 phi	381
6.26.3.28 phi	382
6.26.3.29 phi	382
6.26.3.30 phi	382
6.26.3.31 rho	382
6.26.3.32 rho	382
6.26.3.33 rho	383
6.26.3.34 rho	383
6.26.3.35 vx	383
6.26.3.36 vx	384
6.26.3.37 vxyz	384
6.26.3.38 vxyz	384
6.26.3.39 vy	385
6.26.3.40 vy	385
6.26.3.41 vz	386
6.26.3.42 vz	386
6.26.4 Member Data Documentation	386
6.26.4.1 operator	386

6.27	magrathea::GridCategory Exception Reference	386
6.27.1	Detailed Description	387
6.27.2	Member Function Documentation	387
6.27.2.1	example	387
6.28	Hmaps Class Reference	387
6.28.1	Member Function Documentation	388
6.28.1.1	FillMap	388
6.28.1.2	FillMapPropagate	389
6.28.1.3	getPixels_per_cone	389
6.28.1.4	getPixels_per_cone2	390
6.28.1.5	ReadParamFile	390
6.29	magrathea::HyperCube< Dimension, Vector, Scalar > Exception Template Reference	390
6.29.1	Detailed Description	392
6.29.2	Constructor & Destructor Documentation	392
6.29.2.1	HyperCube	392
6.29.3	Member Function Documentation	392
6.29.3.1	example	392
6.29.3.2	extent	392
6.29.3.3	extent	393
6.29.3.4	position	393
6.29.3.5	position	394
6.29.3.6	unit	394
6.29.4	Member Data Documentation	394
6.29.4.1	operator	394
6.30	magrathea::HyperSphere< Dimension, Vector, Scalar > Exception Template Reference	394
6.30.1	Detailed Description	396
6.30.2	Constructor & Destructor Documentation	396
6.30.2.1	HyperSphere	396
6.30.3	Member Function Documentation	396
6.30.3.1	example	396
6.30.3.2	extent	396
6.30.3.3	extent	397
6.30.3.4	position	397
6.30.3.5	position	398
6.30.3.6	unit	398
6.30.4	Member Data Documentation	398
6.30.4.1	operator	398
6.31	Input Exception Reference	398
6.31.1	Detailed Description	401
6.31.2	Member Function Documentation	401

6.31.2.1	acquire	401
6.31.2.2	collide	402
6.31.2.3	constantify	402
6.31.2.4	correct	403
6.31.2.5	correct	403
6.31.2.6	count	404
6.31.2.7	filetree	404
6.31.2.8	homogenize	405
6.31.2.9	homogenize	405
6.31.2.10	import	405
6.31.2.11	importascii	406
6.31.2.12	importfullhdf5	407
6.31.2.13	importhdf5	407
6.31.2.14	load	408
6.31.2.15	mean	408
6.31.2.16	meanAll	409
6.31.2.17	parse	409
6.31.2.18	partition	410
6.31.2.19	prepare	410
6.31.2.20	save	410
6.31.2.21	schwarzschildify	411
6.31.2.22	schwarzschildify	411
6.31.2.23	sistemize	412
6.31.2.24	sistemize	412
6.31.2.25	trim	413
6.32	Integrator Exception Reference	413
6.32.1	Detailed Description	415
6.32.2	Member Function Documentation	415
6.32.2.1	dphotondl	415
6.32.2.2	example	416
6.32.2.3	integrate	416
6.32.2.4	integrate	416
6.32.2.5	launch	417
6.32.2.6	launch	418
6.32.2.7	launch	418
6.32.2.8	launch	419
6.32.2.9	launch	419
6.32.2.10	propagate	420
6.33	magrathea::LagrangianCategory Exception Reference	421
6.33.1	Detailed Description	421

6.33.2 Member Function Documentation	421
6.33.2.1 example	421
6.34 Lensing Class Reference	421
6.34.1 Detailed Description	422
6.34.2 Member Function Documentation	422
6.34.2.1 dbetadtheta	422
6.34.2.2 dbetadtheta	423
6.34.2.3 dbetadtheta_infinitesimal	424
6.34.2.4 dbetadtheta_infinitesimal	424
6.34.2.5 example	425
6.35 Miscellaneous Class Reference	425
6.35.1 Member Function Documentation	426
6.35.1.1 clear_shrink	426
6.35.1.2 correctOctree	427
6.35.1.3 fill_particles_vectors	427
6.35.1.4 fullclear_vector	428
6.35.1.5 GenerateFullskyCones	428
6.35.1.6 GenerateNarrowCones	429
6.35.1.7 getFilesinDir	429
6.35.1.8 getTargets	429
6.35.1.9 loadOctree	430
6.35.1.10 ReadFromCat	430
6.35.1.11 TicketizeFunction	430
6.35.1.12 Tokenize	431
6.35.1.13 VizualizeOctree	431
6.36 magrathea::NArray< Type, Size > Exception Template Reference	432
6.36.1 Detailed Description	433
6.36.2 Constructor & Destructor Documentation	433
6.36.2.1 NArray	433
6.36.2.2 NArray	433
6.36.2.3 NArray	433
6.36.2.4 NArray	434
6.36.3 Member Function Documentation	434
6.36.3.1 example	434
6.36.4 Member Data Documentation	434
6.36.4.1 _data	434
6.36.4.2 operator	434
6.37 Observer_velocity Class Reference	434
6.37.1 Member Function Documentation	435
6.37.1.1 CreateOctreeVelocityWithCIC	435

6.37.1.2 CreateOctreeVelocityWithTSC	436
6.37.1.3 ReadParamFile	436
6.38 Output Exception Reference	436
6.38.1 Detailed Description	437
6.38.2 Member Function Documentation	437
6.38.2.1 example	437
6.38.2.2 name	438
6.38.2.3 name	438
6.38.2.4 name	438
6.38.2.5 name	439
6.38.2.6 save	439
6.38.2.7 save	439
6.38.2.8 save	440
6.38.2.9 save	440
6.39 parameters_t Struct Reference	441
6.39.1 Member Data Documentation	442
6.39.1.1 acorrection	442
6.39.1.2 allocation	442
6.39.1.3 base	442
6.39.1.4 cat_accuracy	442
6.39.1.5 celldir	442
6.39.1.6 cellfmt	442
6.39.1.7 coarsecorrection	442
6.39.1.8 coarseonly	442
6.39.1.9 conedir	442
6.39.1.10 conefmt	442
6.39.1.11 correction	442
6.39.1.12 evolfile	442
6.39.1.13 firstcone	442
6.39.1.14 halos	443
6.39.1.15 inputtype	443
6.39.1.16 isfullsky	443
6.39.1.17 jacobiantype	443
6.39.1.18 lastcone	443
6.39.1.19 makestat	443
6.39.1.20 map_components	443
6.39.1.21 massmax	443
6.39.1.22 massmin	443
6.39.1.23 microcoeff	443
6.39.1.24 minicone	443

6.39.1.25 mpc	443
6.39.1.26 nb_z_maps	443
6.39.1.27 nbundlecnt	443
6.39.1.28 nbundlemin	443
6.39.1.29 ncoarse	443
6.39.1.30 ncones	443
6.39.1.31 npart	443
6.39.1.32 nside	443
6.39.1.33 nstat	443
6.39.1.34 nsteps	443
6.39.1.35 ntrajectories	443
6.39.1.36 openingcnt	443
6.39.1.37 openingmin	443
6.39.1.38 outputdir	443
6.39.1.39 outputprefix	443
6.39.1.40 paramfile	443
6.39.1.41 partdir	443
6.39.1.42 plane	444
6.39.1.43 ray_targets	444
6.39.1.44 rhoch2	444
6.39.1.45 savemode	444
6.39.1.46 seed	444
6.39.1.47 sourcedir	444
6.39.1.48 statistic	444
6.39.1.49 stop_bundle	444
6.39.1.50 stop_ray	444
6.39.1.51 typefile	444
6.39.1.52 use_previous_catalogues	444
6.39.1.53 v0x	444
6.39.1.54 v0y	444
6.39.1.55 v0z	444
6.39.1.56 velocity_field_v0	444
6.39.1.57 z_stop_max	444
6.39.1.58 z_stop_min	444
6.39.1.59 zmax	444
6.39.1.60 zmin	444
6.40 Photon< Type, Dimension > Exception Template Reference	444
6.40.1 Detailed Description	451
6.40.2 Constructor & Destructor Documentation	451
6.40.2.1 Photon	451

6.40.3 Member Function Documentation	451
6.40.3.1 a	451
6.40.3.2 a	452
6.40.3.3 ah	452
6.40.3.4 ah	453
6.40.3.5 chi	453
6.40.3.6 chi	453
6.40.3.7 distance	454
6.40.3.8 distance	454
6.40.3.9 dphidl	454
6.40.3.10 dphidl	455
6.40.3.11 dphidt	455
6.40.3.12 dphidt	456
6.40.3.13 dphidx	456
6.40.3.14 dphidx	456
6.40.3.15 dphidy	457
6.40.3.16 dphidy	457
6.40.3.17 dphidz	457
6.40.3.18 dphidz	458
6.40.3.19 dsdl2	458
6.40.3.20 dsdl2	459
6.40.3.21 dtdl	459
6.40.3.22 dtdl	459
6.40.3.23 dxdl	460
6.40.3.24 dxdl	460
6.40.3.25 dydl	460
6.40.3.26 dydl	461
6.40.3.27 dzdl	461
6.40.3.28 dzdl	462
6.40.3.29 error	462
6.40.3.30 error	462
6.40.3.31 example	463
6.40.3.32 index	463
6.40.3.33 index	463
6.40.3.34 isw	464
6.40.3.35 isw	464
6.40.3.36 iswold	464
6.40.3.37 iswold	465
6.40.3.38 lambda	465
6.40.3.39 lambda	466

6.40.3.40 laplacian	466
6.40.3.41 laplacian	466
6.40.3.42 level	467
6.40.3.43 level	467
6.40.3.44 phi	467
6.40.3.45 phi	468
6.40.3.46 redshift	468
6.40.3.47 redshift	469
6.40.3.48 rho	469
6.40.3.49 rho	469
6.40.3.50 s	470
6.40.3.51 s	470
6.40.3.52 t	470
6.40.3.53 t	471
6.40.3.54 x	471
6.40.3.55 x	472
6.40.3.56 y	472
6.40.3.57 y	472
6.40.3.58 z	473
6.40.3.59 z	473
6.40.4 Member Data Documentation	473
6.40.4.1 operator	473
6.41 Rays Class Reference	474
6.41.1 Member Function Documentation	474
6.41.1.1 ReadParamFile	474
6.42 magrathea::Shape Exception Reference	474
6.42.1 Detailed Description	475
6.42.2 Member Function Documentation	475
6.42.2.1 example	475
6.43 magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > Exception Template Reference	475
6.43.1 Detailed Description	482
6.43.2 Constructor & Destructor Documentation	482
6.43.2.1 SimpleHyperOctree	482
6.43.2.2 SimpleHyperOctree	483
6.43.3 Member Function Documentation	483
6.43.3.1 a	483
6.43.3.2 a	483
6.43.3.3 append	484
6.43.3.4 assign	484

6.43.3.5 assign	484
6.43.3.6 assign	485
6.43.3.7 at	485
6.43.3.8 at	485
6.43.3.9 back	486
6.43.3.10 back	486
6.43.3.11 begin	486
6.43.3.12 begin	486
6.43.3.13 capacity	487
6.43.3.14 cast	487
6.43.3.15 cbegin	487
6.43.3.16 cend	488
6.43.3.17 cic	488
6.43.3.18 clear	488
6.43.3.19 coarsen	489
6.43.3.20 container	489
6.43.3.21 container	489
6.43.3.22 copy	489
6.43.3.23 crbegin	490
6.43.3.24 crend	490
6.43.3.25 cycle	490
6.43.3.26 cycle	490
6.43.3.27 data	491
6.43.3.28 data	491
6.43.3.29 dimension	491
6.43.3.30 dphidx	491
6.43.3.31 dphidx	492
6.43.3.32 dphidxyz	492
6.43.3.33 dphidxyz	493
6.43.3.34 dphidy	493
6.43.3.35 dphidy	493
6.43.3.36 dphidz	494
6.43.3.37 dphidz	494
6.43.3.38 element	495
6.43.3.39 empty	495
6.43.3.40 end	495
6.43.3.41 end	495
6.43.3.42 extent	496
6.43.3.43 find	496
6.43.3.44 find	496

6.43.3.45 front	496
6.43.3.46 front	496
6.43.3.47 fullclear	497
6.43.3.48 index	497
6.43.3.49 leaf	497
6.43.3.50 locate	498
6.43.3.51 locate	498
6.43.3.52 ngp	498
6.43.3.53 nullify	498
6.43.3.54 operator!=	499
6.43.3.55 operator()	499
6.43.3.56 operator()	499
6.43.3.57 operator()	499
6.43.3.58 operator()	500
6.43.3.59 operator()	500
6.43.3.60 operator()	500
6.43.3.61 operator==	501
6.43.3.62 operator[]	501
6.43.3.63 operator[]	501
6.43.3.64 phi	502
6.43.3.65 phi	502
6.43.3.66 pop	502
6.43.3.67 position	503
6.43.3.68 rbegin	503
6.43.3.69 rbegin	503
6.43.3.70 refine	503
6.43.3.71 rend	504
6.43.3.72 rend	504
6.43.3.73 reserve	504
6.43.3.74 resize	505
6.43.3.75 rho	505
6.43.3.76 rho	506
6.43.3.77 root	506
6.43.3.78 shrink	506
6.43.3.79 size	507
6.43.3.80 space	507
6.43.3.81 tsc	507
6.43.3.82 tsc	507
6.43.3.83 type	508
6.43.3.84 update	508

6.43.4 Friends And Related Function Documentation	508
6.43.4.1 operator<<	508
6.44 magrathea::SimpleHyperOctreeIndex< Type, Dimension, Bits > Exception Template Reference	509
6.44.1 Detailed Description	511
6.44.2 Constructor & Destructor Documentation	511
6.44.2.1 SimpleHyperOctreeIndex	512
6.44.2.2 SimpleHyperOctreeIndex	512
6.44.2.3 SimpleHyperOctreeIndex	512
6.44.3 Member Function Documentation	512
6.44.3.1 assign	512
6.44.3.2 assign	512
6.44.3.3 assign	513
6.44.3.4 assign	513
6.44.3.5 brother	513
6.44.3.6 cast	514
6.44.3.7 check	514
6.44.3.8 child	514
6.44.3.9 coarsest	514
6.44.3.10 copy	515
6.44.3.11 data	515
6.44.3.12 data	515
6.44.3.13 finest	515
6.44.3.14 fix	515
6.44.3.15 following	516
6.44.3.16 get	516
6.44.3.17 invalidate	516
6.44.3.18 invalidated	516
6.44.3.19 level	516
6.44.3.20 limited	517
6.44.3.21 next	517
6.44.3.22 nullify	517
6.44.3.23 operator Type	517
6.44.3.24 operator()	517
6.44.3.25 operator()	518
6.44.3.26 operator=	518
6.44.3.27 operator=	518
6.44.3.28 operator[]	518
6.44.3.29 parent	519
6.44.3.30 preceding	519
6.44.3.31 previous	519

6.44.3.32 <code>set</code>	520
6.44.3.33 <code>stringify</code>	520
6.45 <code>magrathea::StaticVector< Type, Size ></code> Exception Template Reference	520
6.45.1 Detailed Description	522
6.45.2 Constructor & Destructor Documentation	522
6.45.2.1 <code>StaticVector</code>	522
6.45.2.2 <code>StaticVector</code>	522
6.45.2.3 <code>StaticVector</code>	522
6.45.2.4 <code>StaticVector</code>	523
6.45.3 Member Function Documentation	523
6.45.3.1 <code>example</code>	523
6.45.3.2 <code>operator[]</code>	523
6.45.3.3 <code>operator[]</code>	523
6.45.4 Member Data Documentation	524
6.45.4.1 <code>_data</code>	524
6.45.4.2 <code>operator</code>	524
6.46 <code>magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters ></code> Class Template Reference	524
6.46.1 Detailed Description	535
6.46.2 Constructor & Destructor Documentation	536
6.46.2.1 <code>~StaticVectorizer</code>	536
6.46.3 Member Function Documentation	536
6.46.3.1 <code>all</code>	536
6.46.3.2 <code>all</code>	536
6.46.3.3 <code>any</code>	537
6.46.3.4 <code>any</code>	537
6.46.3.5 <code>apply</code>	538
6.46.3.6 <code>apply</code>	538
6.46.3.7 <code>assign</code>	539
6.46.3.8 <code>assign</code>	539
6.46.3.9 <code>at</code>	540
6.46.3.10 <code>at</code>	540
6.46.3.11 <code>back</code>	540
6.46.3.12 <code>back</code>	541
6.46.3.13 <code>boolean</code>	541
6.46.3.14 <code>bytes</code>	541
6.46.3.15 <code>capacity</code>	541
6.46.3.16 <code>cast</code>	542
6.46.3.17 <code>change</code>	542
6.46.3.18 <code>clear</code>	542
6.46.3.19 <code>combine</code>	543

6.46.3.20 combine	543
6.46.3.21 constant	543
6.46.3.22 copy	544
6.46.3.23 count	544
6.46.3.24 count	544
6.46.3.25 cycle	545
6.46.3.26 cycle	545
6.46.3.27 empty	545
6.46.3.28 eq	546
6.46.3.29 example	546
6.46.3.30 fill	546
6.46.3.31 fill	547
6.46.3.32 front	547
6.46.3.33 front	547
6.46.3.34 mask	548
6.46.3.35 mask	548
6.46.3.36 mask	548
6.46.3.37 mask	549
6.46.3.38 max	549
6.46.3.39 min	550
6.46.3.40 modify	550
6.46.3.41 modify	551
6.46.3.42 ne	551
6.46.3.43 none	552
6.46.3.44 none	552
6.46.3.45 null	553
6.46.3.46 nullify	553
6.46.3.47 operator!	553
6.46.3.48 operator!=	553
6.46.3.49 operator!=	554
6.46.3.50 operator%	554
6.46.3.51 operator%	555
6.46.3.52 operator%=	555
6.46.3.53 operator&	555
6.46.3.54 operator&	556
6.46.3.55 operator&&	556
6.46.3.56 operator&&	556
6.46.3.57 operator&=	557
6.46.3.58 operator()	557
6.46.3.59 operator()	557

6.46.3.60 operator()	558
6.46.3.61 operator()	558
6.46.3.62 operator*	558
6.46.3.63 operator*	559
6.46.3.64 operator*= <td>559</td>	559
6.46.3.65 operator+ <td>559</td>	559
6.46.3.66 operator+ <td>560</td>	560
6.46.3.67 operator+ <td>560</td>	560
6.46.3.68 operator++ <td>560</td>	560
6.46.3.69 operator++ <td>561</td>	561
6.46.3.70 operator+= <td>561</td>	561
6.46.3.71 operator- <td>561</td>	561
6.46.3.72 operator- <td>562</td>	562
6.46.3.73 operator- <td>562</td>	562
6.46.3.74 operator-- <td>562</td>	562
6.46.3.75 operator-- <td>562</td>	562
6.46.3.76 operator-= <td>563</td>	563
6.46.3.77 operator/ <td>563</td>	563
6.46.3.78 operator/ <td>563</td>	563
6.46.3.79 operator/= <td>564</td>	564
6.46.3.80 operator< <td>564</td>	564
6.46.3.81 operator< <td>565</td>	565
6.46.3.82 operator<< <td>565</td>	565
6.46.3.83 operator<< <td>565</td>	565
6.46.3.84 operator<=< <td>566</td>	566
6.46.3.85 operator<=< <td>566</td>	566
6.46.3.86 operator<=< <td>566</td>	566
6.46.3.87 operator= <td>567</td>	567
6.46.3.88 operator= <td>567</td>	567
6.46.3.89 operator= <td>568</td>	568
6.46.3.90 operator== <td>568</td>	568
6.46.3.91 operator== <td>568</td>	568
6.46.3.92 operator> <td>569</td>	569
6.46.3.93 operator> <td>569</td>	569
6.46.3.94 operator>= <td>570</td>	570
6.46.3.95 operator>= <td>570</td>	570
6.46.3.96 operator>> <td>570</td>	570
6.46.3.97 operator>> <td>571</td>	571
6.46.3.98 operator>=> <td>571</td>	571
6.46.3.99 operator[] <td>571</td>	571

6.46.3.100operator[]	572
6.46.3.101operator^	572
6.46.3.102operator^=	572
6.46.3.103operator^=	573
6.46.3.104operator 	573
6.46.3.105operator 	574
6.46.3.106operator =	574
6.46.3.107operator 	574
6.46.3.108operator 	575
6.46.3.109operator~	575
6.46.3.110parameters	575
6.46.3.111put	576
6.46.3.112reduce	576
6.46.3.113reduce	576
6.46.3.114replace	577
6.46.3.115replace	577
6.46.3.116reserve	578
6.46.3.117resize	578
6.46.3.118size	578
6.46.3.119space	579
6.46.3.120swap	579
6.46.3.121bytes	579
6.46.3.122type	579
6.46.4 Friends And Related Function Documentation	580
6.46.4.1 operator!=	580
6.46.4.2 operator%	580
6.46.4.3 operator&	581
6.46.4.4 operator&&	581
6.46.4.5 operator*	582
6.46.4.6 operator+	582
6.46.4.7 operator-	583
6.46.4.8 operator/	583
6.46.4.9 operator<	584
6.46.4.10 operator<<	584
6.46.4.11 operator<<<	585
6.46.4.12 operator<=	585
6.46.4.13 operator==	586
6.46.4.14 operator>	586
6.46.4.15 operator>=	587
6.46.4.16 operator>>	587

6.46.4.17 operator>>	588
6.46.4.18 operator^	588
6.46.4.19 operator 	589
6.46.4.20 operator 	589
6.47 magrathea::Step< Scalar, Array, Tuple > Exception Template Reference	590
6.47.1 Detailed Description	590
6.47.2 Constructor & Destructor Documentation	591
6.47.2.1 Step	591
6.47.3 Member Function Documentation	591
6.47.3.1 example	591
6.47.4 Member Data Documentation	591
6.47.4.1 operator	591
6.48 magrathea::Substance< Types > Exception Template Reference	591
6.48.1 Detailed Description	592
6.48.2 Constructor & Destructor Documentation	592
6.48.2.1 Substance	592
6.48.3 Member Function Documentation	593
6.48.3.1 example	593
6.48.4 Member Data Documentation	593
6.48.4.1 operator	593
6.49 magrathea::Timer< Type, Period, Clock > Exception Template Reference	593
6.49.1 Detailed Description	595
6.49.2 Constructor & Destructor Documentation	596
6.49.2.1 Timer	596
6.49.2.2 Timer	596
6.49.3 Member Function Documentation	596
6.49.3.1 assign	596
6.49.3.2 assign	597
6.49.3.3 beginning	597
6.49.3.4 benchmark	598
6.49.3.5 cast	598
6.49.3.6 copy	598
6.49.3.7 current	599
6.49.3.8 ending	599
6.49.3.9 example	599
6.49.3.10 operator()	599
6.49.3.11 operator=	599
6.49.3.12 real	600
6.49.3.13 record	600
6.49.3.14 reference	600

6.49.3.15 reset	600
6.49.3.16 running	601
6.49.3.17 start	601
6.49.3.18 stop	601
6.49.3.19 total	601
6.49.3.20 wait	601
6.49.4 Friends And Related Function Documentation	602
6.49.4.1 operator<<	602
6.49.5 Member Data Documentation	602
6.49.5.1 _beginning	602
6.49.5.2 _ending	602
6.49.5.3 _record	603
6.49.5.4 _reference	603
6.49.5.5 _running	603
6.50 TReadHDF5 Class Reference	603
6.50.1 Member Function Documentation	605
6.50.1.1 cellsAndCubesPerLevels	605
6.50.1.2 cellsPerLevels	606
6.50.1.3 displayAllInfos	606
6.50.1.4 fillVectors_grav	606
6.50.1.5 fillVectors_grav	607
6.50.1.6 fillVectors_grav	607
6.50.1.7 fillVectors_grav	607
6.50.1.8 fillVectors_part	608
6.50.1.9 fillVectors_part	608
6.50.1.10 fillVectors_part	608
6.50.1.11 fillVectors_part	609
6.50.1.12 fillVectors_part	609
6.50.1.13 fillVectors_part	609
6.50.1.14 fillVectors_part	610
6.50.1.15 fillVectors_part	610
6.50.1.16 get_data_from_dataset	611
6.50.1.17 get_data_from_dataset	611
6.50.1.18 getAttribute	611
6.50.1.19 getAttribute	612
6.50.1.20 getAttribute	612
6.50.1.21 getAttribute	612
6.50.1.22 h5t_native	613
6.50.1.23 h5t_native	613
6.50.1.24 h5t_native	613

6.50.1.25 h5t_native	613
6.50.1.26 h5t_native	614
6.50.1.27 h5t_native	614
6.50.1.28 pos_from_index_fullsky	614
6.50.1.29 pos_from_index_narrow	615
6.51 Utility Exception Reference	615
6.51.1 Detailed Description	617
6.51.2 Member Function Documentation	618
6.51.2.1 apply	618
6.51.2.2 collide	618
6.51.2.3 collide	618
6.51.2.4 cross	619
6.51.2.5 cubify	619
6.51.2.6 derive	620
6.51.2.7 differentiate	620
6.51.2.8 distance	621
6.51.2.9 dot	621
6.51.2.10 example	621
6.51.2.11 filter	622
6.51.2.12 integrate	622
6.51.2.13 interpolate	622
6.51.2.14 interpolate	623
6.51.2.15 interpolate2	623
6.51.2.16 invMatrix2d	624
6.51.2.17 invMatrix3d	624
6.51.2.18 join	624
6.51.2.19 parallelize	625
6.51.2.20 parallelize	625
6.51.2.21 parallelize	626
6.51.2.22 radius	626
6.51.2.23 radius	626
6.51.2.24 reinterpolate	627
6.51.2.25 reverse	627
6.51.2.26 smooth	628
6.51.2.27 spherify	628
6.52 magrathea::Vectorized< Type, Size > Exception Template Reference	628
6.52.1 Detailed Description	630
6.52.2 Constructor & Destructor Documentation	630
6.52.2.1 Vectorized	630
6.52.2.2 Vectorized	630

6.52.2.3	Vectorized	631
6.52.2.4	Vectorized	631
6.52.3	Member Function Documentation	631
6.52.3.1	boolean	631
6.52.3.2	capacity	631
6.52.3.3	constant	631
6.52.3.4	example	632
6.52.3.5	operator[]	632
6.52.3.6	operator[]	632
6.52.3.7	parameters	632
6.52.3.8	reserve	633
6.52.3.9	resize	633
6.52.3.10	size	633
6.52.3.11	type	633
6.52.4	Friends And Related Function Documentation	633
6.52.4.1	operator<<	633
6.52.5	Member Data Documentation	634
6.52.5.1	_data	634
6.53	magrathea::Vectorizer Class Reference	634
6.53.1	Detailed Description	636
6.53.2	Constructor & Destructor Documentation	637
6.53.2.1	~Vectorizer	637
6.53.3	Member Function Documentation	637
6.53.3.1	boolean	637
6.53.3.2	check	637
6.53.3.3	check	638
6.53.3.4	check	638
6.53.3.5	constant	638
6.53.3.6	example	638
6.53.3.7	get	639
6.53.3.8	get	639
6.53.3.9	get	639
6.53.3.10	get	640
6.53.3.11	get	640
6.53.3.12	operator[]	640
6.53.3.13	operator[]	641
6.53.3.14	parameters	641
6.53.3.15	resize	641
6.53.3.16	set	642
6.53.3.17	set	642

6.53.3.18 set	642
6.53.3.19 set	643
6.53.3.20 set	643
6.53.3.21 set	644
6.53.3.22 set	644
6.53.3.23 set	645
6.53.3.24 set	645
6.53.3.25 size	646
6.53.3.26 type	646
6.54 magrathea::Wrapper< Type > Exception Template Reference	646
6.54.1 Detailed Description	647
6.54.2 Constructor & Destructor Documentation	648
6.54.2.1 Wrapper	648
6.54.2.2 Wrapper	648
6.54.3 Member Function Documentation	648
6.54.3.1 example	648
6.54.3.2 operator Type	648
6.54.3.3 operator()	648
6.54.3.4 operator()	649
6.54.3.5 operator()	649
6.54.3.6 operator()	649
6.54.3.7 operator=	650
6.54.3.8 operator=	650
6.54.4 Member Data Documentation	650
6.54.4.1 _data	650
7 File Documentation	651
7.1 /data/home/mbreton/magrathea_raytracing/src/catalogues.cpp File Reference	651
7.1.1 Function Documentation	652
7.1.1.1 main	652
7.2 /data/home/mbreton/magrathea_raytracing/src/catalogues.h File Reference	652
7.2.1 Detailed Description	653
7.2.2 Variable Documentation	653
7.2.2.1 parameters	653
7.3 /data/home/mbreton/magrathea_raytracing/src/cone.h File Reference	653
7.3.1 Detailed Description	653
7.4 /data/home/mbreton/magrathea_raytracing/src/create_octree.cpp File Reference	654
7.4.1 Function Documentation	654
7.4.1.1 main	654
7.5 /data/home/mbreton/magrathea_raytracing/src/create_octree.h File Reference	655

7.5.1	Detailed Description	655
7.5.2	Variable Documentation	656
7.5.2.1	parameters	656
7.6	/data/home/mbreton/magrathea_raytracing/src/gravity.h File Reference	656
7.6.1	Detailed Description	656
7.7	/data/home/mbreton/magrathea_raytracing/src/gravity2.h File Reference	657
7.7.1	Detailed Description	657
7.8	/data/home/mbreton/magrathea_raytracing/src/hmaps.cpp File Reference	658
7.8.1	Function Documentation	658
7.8.1.1	main	658
7.9	/data/home/mbreton/magrathea_raytracing/src/hmaps.h File Reference	659
7.9.1	Detailed Description	660
7.9.2	Variable Documentation	660
7.9.2.1	parameters	660
7.10	/data/home/mbreton/magrathea_raytracing/src/input.h File Reference	660
7.10.1	Detailed Description	661
7.11	/data/home/mbreton/magrathea_raytracing/src/integrator.h File Reference	661
7.11.1	Detailed Description	662
7.12	/data/home/mbreton/magrathea_raytracing/src/lensing.h File Reference	662
7.12.1	Detailed Description	662
7.13	/data/home/mbreton/magrathea_raytracing/src/magrathea/aboutinstitute.h File Reference	663
7.13.1	Detailed Description	663
7.14	/data/home/mbreton/magrathea_raytracing/src/magrathea/aboutlicense.h File Reference	664
7.14.1	Detailed Description	664
7.15	/data/home/mbreton/magrathea_raytracing/src/magrathea/aboutobject.h File Reference	664
7.15.1	Detailed Description	665
7.16	/data/home/mbreton/magrathea_raytracing/src/magrathea/aboutpeople.h File Reference	665
7.16.1	Detailed Description	665
7.17	/data/home/mbreton/magrathea_raytracing/src/magrathea/abstractaboutobject.h File Reference	666
7.17.1	Detailed Description	666
7.18	/data/home/mbreton/magrathea_raytracing/src/magrathea/abstractcontents.h File Reference	667
7.18.1	Detailed Description	667
7.19	/data/home/mbreton/magrathea_raytracing/src/magrathea/abstracthypercube.h File Reference	668
7.19.1	Detailed Description	668
7.20	/data/home/mbreton/magrathea_raytracing/src/magrathea/abstracthypersphere.h File Reference	668
7.20.1	Detailed Description	669
7.21	/data/home/mbreton/magrathea_raytracing/src/magrathea/abstractnarray.h File Reference	669
7.21.1	Detailed Description	670
7.22	/data/home/mbreton/magrathea_raytracing/src/magrathea/abstractshape.h File Reference	670
7.22.1	Detailed Description	670

7.23 /data/home/mbreton/magrathea_raytracing/src/magrathea/abstractstep.h File Reference	671
7.23.1 Detailed Description	671
7.24 /data/home/mbreton/magrathea_raytracing/src/magrathea/abstractsubstance.h File Reference	671
7.24.1 Detailed Description	672
7.25 /data/home/mbreton/magrathea_raytracing/src/magrathea/arrayfile.h File Reference	672
7.25.1 Detailed Description	673
7.26 /data/home/mbreton/magrathea_raytracing/src/magrathea/basefile.h File Reference	673
7.26.1 Detailed Description	673
7.27 /data/home/mbreton/magrathea_raytracing/src/magrathea/binaryfile.h File Reference	673
7.27.1 Detailed Description	674
7.28 /data/home/mbreton/magrathea_raytracing/src/magrathea/constant.h File Reference	674
7.28.1 Detailed Description	674
7.29 /data/home/mbreton/magrathea_raytracing/src/magrathea/constants.h File Reference	675
7.29.1 Detailed Description	675
7.30 /data/home/mbreton/magrathea_raytracing/src/magrathea/contents.h File Reference	675
7.30.1 Detailed Description	676
7.31 /data/home/mbreton/magrathea_raytracing/src/magrathea/datahandler.h File Reference	676
7.31.1 Detailed Description	677
7.32 /data/home/mbreton/magrathea_raytracing/src/magrathea/datamodel.h File Reference	677
7.32.1 Detailed Description	678
7.33 /data/home/mbreton/magrathea_raytracing/src/magrathea/datasize.h File Reference	678
7.33.1 Detailed Description	679
7.34 /data/home/mbreton/magrathea_raytracing/src/magrathea/euleriancategory.h File Reference	679
7.34.1 Detailed Description	679
7.35 /data/home/mbreton/magrathea_raytracing/src/magrathea/evolution.h File Reference	680
7.35.1 Detailed Description	680
7.36 /data/home/mbreton/magrathea_raytracing/src/magrathea/filelist.h File Reference	680
7.36.1 Detailed Description	681
7.37 /data/home/mbreton/magrathea_raytracing/src/magrathea/filesystem.h File Reference	681
7.37.1 Detailed Description	682
7.38 /data/home/mbreton/magrathea_raytracing/src/magrathea/fortranfile.h File Reference	682
7.38.1 Detailed Description	683
7.39 /data/home/mbreton/magrathea_raytracing/src/magrathea/gridcategory.h File Reference	683
7.39.1 Detailed Description	683
7.40 /data/home/mbreton/magrathea_raytracing/src/magrathea/hypercube.h File Reference	684
7.40.1 Detailed Description	684
7.41 /data/home/mbreton/magrathea_raytracing/src/magrathea/hypersphere.h File Reference	684
7.41.1 Detailed Description	685
7.42 /data/home/mbreton/magrathea_raytracing/src/magrathea/lagrangiancategory.h File Reference	685
7.42.1 Detailed Description	685

7.43 /data/home/mbreton/magrathea_raytracing/src/magrathea/logfile.h File Reference	686
7.43.1 Detailed Description	686
7.44 /data/home/mbreton/magrathea_raytracing/src/magrathea/namelistfile.h File Reference	686
7.44.1 Detailed Description	687
7.45 /data/home/mbreton/magrathea_raytracing/src/magrathea/narray.h File Reference	687
7.45.1 Detailed Description	687
7.46 /data/home/mbreton/magrathea_raytracing/src/magrathea/shape.h File Reference	688
7.46.1 Detailed Description	688
7.47 /data/home/mbreton/magrathea_raytracing/src/magrathea/simplehyperoctree.h File Reference	688
7.47.1 Detailed Description	689
7.48 /data/home/mbreton/magrathea_raytracing/src/magrathea/simplehyperoctreeindex.h File Reference	689
7.48.1 Detailed Description	690
7.49 /data/home/mbreton/magrathea_raytracing/src/magrathea/staticvector.h File Reference	690
7.49.1 Detailed Description	691
7.50 /data/home/mbreton/magrathea_raytracing/src/magrathea/staticvectorizer.h File Reference	691
7.50.1 Detailed Description	694
7.51 /data/home/mbreton/magrathea_raytracing/src/magrathea/step.h File Reference	695
7.51.1 Detailed Description	695
7.52 /data/home/mbreton/magrathea_raytracing/src/magrathea/substance.h File Reference	696
7.52.1 Detailed Description	696
7.53 /data/home/mbreton/magrathea_raytracing/src/magrathea/temporaryfile.h File Reference	696
7.53.1 Detailed Description	697
7.54 /data/home/mbreton/magrathea_raytracing/src/magrathea/textfile.h File Reference	697
7.54.1 Detailed Description	697
7.55 /data/home/mbreton/magrathea_raytracing/src/magrathea/timer.h File Reference	698
7.55.1 Detailed Description	698
7.56 /data/home/mbreton/magrathea_raytracing/src/magrathea/vectorized.h File Reference	698
7.56.1 Detailed Description	699
7.57 /data/home/mbreton/magrathea_raytracing/src/magrathea/vectorizer.h File Reference	699
7.57.1 Detailed Description	700
7.58 /data/home/mbreton/magrathea_raytracing/src/magrathea(wrapper.h File Reference	700
7.58.1 Detailed Description	700
7.59 /data/home/mbreton/magrathea_raytracing/src/miscellaneous.h File Reference	701
7.59.1 Detailed Description	702
7.59.2 Typedef Documentation	702
7.59.2.1 floating	702
7.59.2.2 integer	702
7.59.2.3 point	702
7.59.2.4 real	702
7.59.2.5 uint	702

7.60 /data/home/mbreton/magrathea_raytracing/src/observer_velocity.cpp File Reference	703
7.60.1 Function Documentation	703
7.60.1.1 main	703
7.61 /data/home/mbreton/magrathea_raytracing/src/observer_velocity.h File Reference	704
7.61.1 Detailed Description	704
7.61.2 Macro Definition Documentation	705
7.61.2.1 OBSEREVR_VELOCITY_H_INCLUDED	705
7.61.3 Variable Documentation	705
7.61.3.1 parameters	705
7.62 /data/home/mbreton/magrathea_raytracing/src/output.h File Reference	705
7.62.1 Detailed Description	705
7.63 /data/home/mbreton/magrathea_raytracing/src/photon.h File Reference	706
7.63.1 Detailed Description	706
7.64 /data/home/mbreton/magrathea_raytracing/src/rays.cpp File Reference	707
7.64.1 Function Documentation	707
7.64.1.1 main	707
7.65 /data/home/mbreton/magrathea_raytracing/src/rays.h File Reference	708
7.65.1 Detailed Description	708
7.65.2 Variable Documentation	709
7.65.2.1 parameters	709
7.66 /data/home/mbreton/magrathea_raytracing/src/TReadHDF5.h File Reference	709
7.66.1 Detailed Description	709
7.67 /data/home/mbreton/magrathea_raytracing/src/utility.h File Reference	710
7.67.1 Detailed Description	710
Index	710

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

magrathea	13
-----------	-------	----

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

magrathea::AbstractAboutObject< Crtp, Types >	63
magrathea::AbstractAboutObject< AboutInstitute, std::string, std::string, std::string, std::string, std::string, std::string, std::string, std::string, std::string, std::string >	63
magrathea::AboutInstitute	35
magrathea::AbstractAboutObject< AboutLicense, std::string, std::string, std::string, bool, double, std::string, std::string, std::string >	63
magrathea::AboutLicense	45
magrathea::AbstractAboutObject< AboutObject< Types...>, Types...>	63
magrathea::AboutObject< Types >	54
magrathea::AbstractAboutObject< AboutPeople, std::string, std::string, int, int, std::string, std::string, std::string, std::string >	63
magrathea::AboutPeople	55
magrathea::AbstractContents< Crtp, Category, Types >	72
magrathea::AbstractContents< Contents< Category, Types...>, Category, Types...>	72
magrathea::Contents< Category, Types >	221
magrathea::AbstractContents< Gravity< Type, Dimension >, magrathea::EulerianCategory, Type, Type, std::array< Type, Dimension >, Type, Type >	72
Gravity< Type, Dimension >	370
magrathea::AbstractContents< Gravity< Type, Dimension >, magrathea::EulerianCategory, Type, Type, std::array< Type, Dimension >, Type, Type, std::array< Type, Dimension > >	72
Gravity< Type, Dimension >	370
magrathea::AbstractShape	124
magrathea::AbstractHyperCube< HyperCube< Dimension, Vector, Scalar >, Dimension, Vector, Scalar >	81
magrathea::HyperCube< Dimension, Vector, Scalar >	390
magrathea::AbstractHyperSphere< HyperSphere< Dimension, Vector, Scalar >, Dimension, Vector, Scalar >	89
magrathea::HyperSphere< Dimension, Vector, Scalar >	394
Cone< Vector, Scalar >	151
magrathea::AbstractHyperCube< Crtp, Dimension, Vector, Scalar >	81
magrathea::AbstractHyperSphere< Crtp, Dimension, Vector, Scalar >	89
magrathea::Shape	474
magrathea::AbstractStep< Crtp, Scalar, Array, Tuple >	129
magrathea::AbstractStep< Photon< Type, Dimension >, unsigned int, std::array< Type, 1+(1+Dimension)*2 >, std::tuple< Type, Type, Type, Type, std::array< Type, Dimension >, Type, Type, Type, Type, std::array< Type, 3 >, Type, Type, Type > >	129

Photon< Type, Dimension >	444
magrathea::AbstractStep< Step< Scalar, Array, Tuple >, Scalar, Array, Tuple >	129
magrathea::Step< Scalar, Array, Tuple >	590
magrathea::AbstractSubstance< Crtp, Types >	138
magrathea::AbstractSubstance< Cone< Vector, Scalar >, Vector, Vector, Scalar >	138
Cone< Vector, Scalar >	151
magrathea::AbstractSubstance< HyperCube< Dimension, Vector, Scalar >, Vector, Scalar >	138
magrathea::HyperCube< Dimension, Vector, Scalar >	390
magrathea::AbstractSubstance< HyperSphere< Dimension, Vector, Scalar >, Vector, Scalar >	138
magrathea::HyperSphere< Dimension, Vector, Scalar >	394
magrathea::AbstractSubstance< Substance< Types...>, Types...>	138
magrathea::Substance< Types >	591
Catalogues	147
magrathea::Constant< Type, Size >	159
magrathea::Constants< Type >	188
Create_octree	223
magrathea::DataHandler	230
magrathea::DataModel	281
magrathea::DataSize	293
magrathea::EulerianCategory	313
magrathea::Evolution< Type, Container >	313
magrathea::FileList	332
magrathea::FileSystem	354
magrathea::GridCategory	386
Hmaps	387
Input	398
Integrator	413
magrathea::LagrangianCategory	421
Lensing	421
Miscellaneous	425
Observer_velocity	434
Output	436
parameters_t	441
Rays	474
magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >	475
magrathea::SimpleHyperOctreeIndex< Type, Dimension, Bits >	509
magrathea::Timer< Type, Period, Clock >	593
TReadHDF5	603
Utility	615
magrathea::Vectorizer	634
magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >	524
magrathea::Vectorized< Type, Size >	628
magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters...>	524
magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >	97
magrathea::StaticVectorizer< unsigned int, Size, NArray, Type, Parameters...>	524
magrathea::AbstractNArray< unsigned int, Size, NArray, Type, Size >	97
magrathea::NArray< Type, Size >	432
magrathea::StaticVectorizer< unsigned int, Size, StaticVector, Type, Size >	524
magrathea::StaticVector< Type, Size >	520
magrathea::Wrapper< Type >	646

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

magrathea::AboutInstitute	Information about an institution or an organization	35
magrathea::AboutLicense	Information about the license of a code	45
magrathea::AboutObject< Types >	Basic about object with information on something	54
magrathea::AboutPeople	Information about a developer, an author, or a contributor	55
magrathea::AbstractAboutObject< Crtp, Types >	Tuple abstraction of generic about object	63
magrathea::AbstractContents< Crtp, Category, Types >	Tuple abstraction of numerical simulation contents	72
magrathea::AbstractHyperCube< Crtp, Dimension, Vector, Scalar >	Abstract function provider for n-dimensional cubes	81
magrathea::AbstractHyperSphere< Crtp, Dimension, Vector, Scalar >	Abstract function provider for n-dimensional spheres	89
magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >	Abstract base class of n-dimensional mathematical arrays	97
magrathea::AbstractShape	Common abstraction of n-dimensional shapes	124
magrathea::AbstractStep< Crtp, Scalar, Array, Tuple >	Abstraction of an evolution step	129
magrathea::AbstractSubstance< Crtp, Types >	Tuple abstraction of geometrical substance	138
Catalogues	147
Cone< Vector, Scalar >	Three-dimensional cone	151
magrathea::Constant< Type, Size >	Numerical constant with constexpr constructor	159
magrathea::Constants< Type >	Common mathematical and physical constants	188
magrathea::Contents< Category, Types >	Basic implementation of numerical simulation contents	221
Create_octree	223
magrathea::DataHandler	Set of basic operations on binary data related to IO	230
magrathea::DataModel	Management of fundamental types representation	281

magrathea::DataSize	
Wrapper of binary data size and manager of unit conversion	293
magrathea::EulerianCategory	
Category concept of eulerian : data at a fixed position	313
magrathea::Evolution< Type, Container >	
Resizable container of steps dedicated to integration	313
magrathea::FileList	
List of files based on a function or a vector	332
magrathea::FileSystem	
Global file management	354
Gravity< Type, Dimension >	
Gravity cell implementation for raytracing	370
magrathea::GridCategory	
Category concept of grid : data related to the mesh	386
Hmaps	
magrathea::HyperCube< Dimension, Vector, Scalar >	
N-dimensional cube	390
magrathea::HyperSphere< Dimension, Vector, Scalar >	
N-dimensional sphere	394
Input	
Input utilities for raytracing	398
Integrator	
Integration utilities for raytracing	413
magrathea::LagrangianCategory	
Category concept of lagrangian : data moving with the flow	421
Lensing	
Integration utilities for raytracing	421
Miscellaneous	
magrathea::NArray< Type, Size >	
Basic n-dimensional mathematical array	432
Observer_velocity	
Output	
Output utilities for raytracing	436
parameters_t	
Photon< Type, Dimension >	
Photon step implementation for raytracing	444
Rays	
magrathea::Shape	
Basic implementation of shape	474
magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >	
A simple hyperoctree based on bit manipulations	475
magrathea::SimpleHyperOctreeIndex< Type, Dimension, Bits >	
A simple hyperoctree index based on an integer	509
magrathea::StaticVector< Type, Size >	
Basic vectorized constant size container	520
magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >	
Helper class for generic constant size vectorization	524
magrathea::Step< Scalar, Array, Tuple >	
Basic implementation of an evolution step	590
magrathea::Substance< Types >	
Basic implementation of geometrical substance	591
magrathea::Timer< Type, Period, Clock >	
A timer to manage time measurements and benchmarks	593
TReadHDF5	
Utility	
List of utilities for raytracing	615
magrathea::Vectorized< Type, Size >	
Basic vectorized container	628

magrathea::Vectorizer	
Helper base class for generic vectorization	634
magrathea::Wrapper< Type >	
Basic value wrapper with getter and setter	646

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

/data/home/mbreton/magrathea_pathfinder/src/ catalogues.cpp	651
Main function of the raytracer to create catalogues	.
/data/home/mbreton/magrathea_pathfinder/src/ catalogues.h	652
Some catalogues function	.
/data/home/mbreton/magrathea_pathfinder/src/ cone.h	653
Three-dimensional cone	.
/data/home/mbreton/magrathea_pathfinder/src/ create_octree.cpp	654
/data/home/mbreton/magrathea_pathfinder/src/ create_octree.h	655
Some create_octree function	.
/data/home/mbreton/magrathea_pathfinder/src/ gravity.h	656
Gravity cell implementation for raytracing	.
/data/home/mbreton/magrathea_pathfinder/src/ gravity2.h	657
Gravity cell implementation for raytracing	.
/data/home/mbreton/magrathea_pathfinder/src/ hmaps.cpp	658
Main function of the raytracer to create Healpix maps	.
/data/home/mbreton/magrathea_pathfinder/src/ hmaps.h	659
Some hmaps function	.
/data/home/mbreton/magrathea_pathfinder/src/ input.h	660
Input utilities for raytracing	.
/data/home/mbreton/magrathea_pathfinder/src/ integrator.h	661
Integration utilities for raytracing	.
/data/home/mbreton/magrathea_pathfinder/src/ lensing.h	662
Integration utilities for raytracing	.
/data/home/mbreton/magrathea_pathfinder/src/ miscellaneous.h	701
Some miscellaneous function	.
/data/home/mbreton/magrathea_pathfinder/src/ observer_velocity.cpp	703
Main function of the raytracer to compute the observer's velocity	.
/data/home/mbreton/magrathea_pathfinder/src/ observer_velocity.h	704
Some observer_velocity function	.
/data/home/mbreton/magrathea_pathfinder/src/ output.h	705
Output utilities for raytracing	.
/data/home/mbreton/magrathea_pathfinder/src/ photon.h	706
Photon step implementation for raytracing	.
/data/home/mbreton/magrathea_pathfinder/src/ rays.cpp	707
Main function of the raytracer	.
/data/home/mbreton/magrathea_pathfinder/src/ rays.h	708
Some rays function	.

/data/home/mbreton/magrathea_pathfinder/src/TReadHDF5.h	Utilities to read HDF5 files	709
/data/home/mbreton/magrathea_pathfinder/src/utility.h	List of utilities for raytracing	710
/data/home/mbreton/magrathea_pathfinder/src/magrathea/aboutinstitute.h	Information about an institution or an organization	663
/data/home/mbreton/magrathea_pathfinder/src/magrathea/aboutlicense.h	Information about the license of a code	664
/data/home/mbreton/magrathea_pathfinder/src/magrathea/aboutobject.h	Basic about object with information on something	664
/data/home/mbreton/magrathea_pathfinder/src/magrathea/aboutpeople.h	Information about a developer, an author, or a contributor	665
/data/home/mbreton/magrathea_pathfinder/src/magrathea/abstractaboutobject.h	Tuple abstraction of generic about object	666
/data/home/mbreton/magrathea_pathfinder/src/magrathea/abstractcontents.h	Tuple abstraction of numerical simulation contents	667
/data/home/mbreton/magrathea_pathfinder/src/magrathea/abstracthypercube.h	Abstract function provider for n-dimensional cubes	668
/data/home/mbreton/magrathea_pathfinder/src/magrathea/abstracthypersphere.h	Abstract function provider for n-dimensional spheres	668
/data/home/mbreton/magrathea_pathfinder/src/magrathea/abstractnarray.h	Abstract base class of n-dimensional mathematical arrays	669
/data/home/mbreton/magrathea_pathfinder/src/magrathea/abstractshape.h	Common abstraction of n-dimensional shapes	670
/data/home/mbreton/magrathea_pathfinder/src/magrathea/abstractstep.h	Abstraction of an evolution step	671
/data/home/mbreton/magrathea_pathfinder/src/magrathea/abstractsubstance.h	Tuple abstraction of geometrical substance	671
/data/home/mbreton/magrathea_pathfinder/src/magrathea/arrayfile.h	Handler for text files representing arrays of data	672
/data/home/mbreton/magrathea_pathfinder/src/magrathea/basefile.h	Base class for input and output with files	673
/data/home/mbreton/magrathea_pathfinder/src/magrathea/binaryfile.h	Base class to manage generic binary files	673
/data/home/mbreton/magrathea_pathfinder/src/magrathea/constant.h	Numerical constant with constexpr constructor	674
/data/home/mbreton/magrathea_pathfinder/src/magrathea/constants.h	Common mathematical and physical constants	675
/data/home/mbreton/magrathea_pathfinder/src/magrathea/contents.h	Basic implementation of numerical simulation contents	675
/data/home/mbreton/magrathea_pathfinder/src/magrathea/datalayer.h	Set of basic operations on binary data related to IO	676
/data/home/mbreton/magrathea_pathfinder/src/magrathea/datamodel.h	Management of fundamental types representation	677
/data/home/mbreton/magrathea_pathfinder/src/magrathea/datasize.h	Wrapper of binary data size and manager of unit conversion	678
/data/home/mbreton/magrathea_pathfinder/src/magrathea/euleriancategory.h	Category concept of eulerian : data at a fixed position	679
/data/home/mbreton/magrathea_pathfinder/src/magrathea/evolution.h	Resizable container of steps dedicated to integration	680
/data/home/mbreton/magrathea_pathfinder/src/magrathea/filelist.h	List of files based on a function or a vector	680
/data/home/mbreton/magrathea_pathfinder/src/magrathea/filesystem.h	Global file management	681
/data/home/mbreton/magrathea_pathfinder/src/magrathea/fortranfile.h	Handler for binary files in fortran format	682
/data/home/mbreton/magrathea_pathfinder/src/magrathea/gridcategory.h	Category concept of grid : data related to the mesh	683

/data/home/mbreton/magrathea_pathfinder/src/magrathea/ hypercube.h	684
N-dimensional cube	684
/data/home/mbreton/magrathea_pathfinder/src/magrathea/ hypersphere.h	684
N-dimensional sphere	684
/data/home/mbreton/magrathea_pathfinder/src/magrathea/ lagrangiancategory.h	685
Category concept of lagrangian : data moving with the flow	685
/data/home/mbreton/magrathea_pathfinder/src/magrathea/ logfile.h	686
Handler for text files representing logs	686
/data/home/mbreton/magrathea_pathfinder/src/magrathea/ namelistfile.h	686
Handler for text files representing lists of parameters	686
/data/home/mbreton/magrathea_pathfinder/src/magrathea/ narray.h	687
Basic n-dimensional mathematical array	687
/data/home/mbreton/magrathea_pathfinder/src/magrathea/ shape.h	688
Basic implementation of shape	688
/data/home/mbreton/magrathea_pathfinder/src/magrathea/ simplehyperoctree.h	688
A simple hyperoctree based on bit manipulations	688
/data/home/mbreton/magrathea_pathfinder/src/magrathea/ simplehyperoctreeindex.h	689
A simple hyperoctree index based on an integer	689
/data/home/mbreton/magrathea_pathfinder/src/magrathea/ staticvector.h	690
Basic vectorized constant size container	690
/data/home/mbreton/magrathea_pathfinder/src/magrathea/ staticvectorizer.h	691
Helper class for generic constant size vectorization	691
/data/home/mbreton/magrathea_pathfinder/src/magrathea/ step.h	695
Basic implementation of an evolution step	695
/data/home/mbreton/magrathea_pathfinder/src/magrathea/ substance.h	696
Basic implementation of geometrical substance	696
/data/home/mbreton/magrathea_pathfinder/src/magrathea/ temporaryfile.h	696
Temporary file with automatic deletion at destruction	696
/data/home/mbreton/magrathea_pathfinder/src/magrathea/ textfile.h	697
Base class to manage generic text files	697
/data/home/mbreton/magrathea_pathfinder/src/magrathea/ timer.h	698
A timer to manage time measurements and benchmarks	698
/data/home/mbreton/magrathea_pathfinder/src/magrathea/ vectorized.h	698
Basic vectorized container	698
/data/home/mbreton/magrathea_pathfinder/src/magrathea/ vectorizer.h	699
Helper base class for generic vectorization	699
/data/home/mbreton/magrathea_pathfinder/src/magrathea/ wrapper.h	700
Basic value wrapper with getter and setter	700

Chapter 5

Namespace Documentation

5.1 magrathea Namespace Reference

Classes

- exception [AboutInstitute](#)
Information about an institution or an organization.
- exception [AboutLicense](#)
Information about the license of a code.
- exception [AboutObject](#)
Basic about object with information on something.
- exception [AboutPeople](#)
Information about a developer, an author, or a contributor.
- class [AbstractAboutObject](#)
Tuple abstraction of generic about object.
- class [AbstractContents](#)
Tuple abstraction of numerical simulation contents.
- class [AbstractHyperCube](#)
Abstract function provider for n-dimensional cubes.
- class [AbstractHyperSphere](#)
Abstract function provider for n-dimensional spheres.
- class [AbstractNArray](#)
Abstract base class of n-dimensional mathematical arrays.
- class [AbstractShape](#)
Common abstraction of n-dimensional shapes.
- class [AbstractStep](#)
Abstraction of an evolution step.
- class [AbstractSubstance](#)
Tuple abstraction of geometrical substance.
- exception [Constant](#)
Numerical constant with constexpr constructor.
- exception [Constants](#)
Common mathematical and physical constants.
- exception [Contents](#)
Basic implementation of numerical simulation contents.
- exception [DataHandler](#)
Set of basic operations on binary data related to IO.

- exception [DataModel](#)

Management of fundamental types representation.
- exception [DataSize](#)

Wrapper of binary data size and manager of unit conversion.
- exception [EulerianCategory](#)

Category concept of eulerian : data at a fixed position.
- exception [Evolution](#)

Resizable container of steps dedicated to integration.
- exception [FileList](#)

List of files based on a function or a vector.
- exception [FileSystem](#)

Global file management.
- exception [GridCategory](#)

Category concept of grid : data related to the mesh.
- exception [HyperCube](#)

N-dimensional cube.
- exception [HyperSphere](#)

N-dimensional sphere.
- exception [LagrangianCategory](#)

Category concept of lagrangian : data moving with the flow.
- exception [NArray](#)

Basic n-dimensional mathematical array.
- exception [Shape](#)

Basic implementation of shape.
- exception [SimpleHyperOctree](#)

A simple hyperoctree based on bit manipulations.
- exception [SimpleHyperOctreeIndex](#)

A simple hyperoctree index based on an integer.
- exception [StaticVector](#)

Basic vectorized constant size container.
- class [StaticVectorizer](#)

Helper class for generic constant size vectorization.
- exception [Step](#)

Basic implementation of an evolution step.
- exception [Substance](#)

Basic implementation of geometrical substance.
- exception [Timer](#)

A timer to manage time measurements and benchmarks.
- exception [Vectorized](#)

Basic vectorized container.
- class [Vectorizer](#)

Helper base class for generic vectorization.
- exception [Wrapper](#)

Basic value wrapper with getter and setter.

Functions

- template<class SelfCrtp , class... SelfTypes>
`std::ostream & operator<< (std::ostream &lhs, const AbstractAboutObject< SelfCrtp, SelfTypes...> &rhs)`
Output stream operator.
- template<class SelfCrtp , class SelfCategory , class... SelfTypes>
`std::ostream & operator<< (std::ostream &lhs, const AbstractContents< SelfCrtp, SelfCategory, SelfTypes...> &rhs)`
Output stream operator.
- template<class SelfCrtp , class SelfScalar , class SelfArray , class SelfTuple >
`std::ostream & operator<< (std::ostream &lhs, const AbstractStep< SelfCrtp, SelfScalar, SelfArray, SelfTuple > &rhs)`
Output stream operator.
- template<class SelfCrtp , class... SelfTypes>
`std::ostream & operator<< (std::ostream &lhs, const AbstractSubstance< SelfCrtp, SelfTypes...> &rhs)`
Output stream operator.
- template<typename SelfType , unsigned int SelfSize>
`std::ostream & operator<< (std::ostream &lhs, const Constant< SelfType, SelfSize > &rhs)`
Output stream operator.
- std::ostream & operator<< (std::ostream &lhs, const DataModel &rhs)
Output stream operator.
- std::ostream & operator<< (std::ostream &lhs, const DataSize &rhs)
Output stream operator.
- template<class SelfType , class SelfContainer >
`std::ostream & operator<< (std::ostream &lhs, const Evolution< SelfType, SelfContainer > &rhs)`
Output stream operator.
- std::ostream & operator<< (std::ostream &lhs, const FileList &rhs)
Output stream operator.
- template<typename SelfType , unsigned int SelfDimension, unsigned int SelfBits>
`std::ostream & operator<< (std::ostream &lhs, const SimpleHyperOctreeIndex< SelfType, SelfDimension, SelfBits > &rhs)`
Output stream operator.
- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, SelfType>::value)>::type>
`SelfCrtp< typename std::common_type< SelfType, OtherType >::type,`
`SelfParameters...> operator+ (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> &rhs)`
Addition with lhs value.
- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, SelfType>::value)>::type>
`SelfCrtp< typename std::common_type< SelfType, OtherType >::type,`
`SelfParameters...> operator- (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> &rhs)`
Subtraction with lhs value.
- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, SelfType>::value)>::type>

```
SelfCrtp< typename
std::common_type< SelfType,
OtherType >::type,
SelfParameters...> operator* (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp,
SelfType, SelfParameters...> &rhs)
```

Multiplication with lhs value.

- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, SelfType>::value)>::type>

```
SelfCrtp< typename
std::common_type< SelfType,
OtherType >::type,
SelfParameters...> operator/ (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp,
SelfType, SelfParameters...> &rhs)
```

Division with lhs value.

- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, SelfType>::value)>::type>

```
SelfCrtp< typename
std::common_type< SelfType,
OtherType >::type,
SelfParameters...> operator% (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp,
SelfType, SelfParameters...> &rhs)
```

Modulo with lhs value.

- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, SelfType>::value)>::type>

```
SelfCrtp< typename
std::common_type< SelfType,
OtherType >::type,
SelfParameters...> operator& (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp,
SelfType, SelfParameters...> &rhs)
```

Bitwise AND with lhs value.

- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, SelfType>::value)>::type>

```
SelfCrtp< typename
std::common_type< SelfType,
OtherType >::type,
SelfParameters...> operator| (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp,
SelfType, SelfParameters...> &rhs)
```

Bitwise OR with lhs value.

- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, SelfType>::value)>::type>

```
SelfCrtp< typename
std::common_type< SelfType,
OtherType >::type,
SelfParameters...> operator^ (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp,
SelfType, SelfParameters...> &rhs)
```

Bitwise XOR with lhs value.

- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, SelfType>::value)>::type>

```
SelfCrtp< OtherType,
SelfParameters...> operator<< (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp,
SelfType, SelfParameters...> &rhs)
```

Bitwise left shift with lhs value.

- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, SelfType>::value)>::type> SelfCrtp< OtherType, SelfParameters...> operator>> (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> &rhs)

Bitwise right shift with lhs value.

- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, SelfType>::value)>::type> SelfCrtp< bool, SelfParameters...> operator&& (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> &rhs)

Logical AND with lhs value.

- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, SelfType>::value)>::type> SelfCrtp< bool, SelfParameters...> operator|| (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> &rhs)

Logical OR with lhs value.

- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, SelfType>::value)>::type> SelfCrtp< bool, SelfParameters...> operator== (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> &rhs)

Equal to with lhs value.

- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, SelfType>::value)>::type> SelfCrtp< bool, SelfParameters...> operator!= (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> &rhs)

Not equal to with lhs value.

- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, SelfType>::value)>::type> SelfCrtp< bool, SelfParameters...> operator> (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> &rhs)

Greater than with lhs value.

- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, SelfType>::value)>::type> SelfCrtp< bool, SelfParameters...> operator< (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> &rhs)

Less than with lhs value.

- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, SelfType>::value)>::type> SelfCrtp< bool, SelfParameters...> operator>= (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> &rhs)

Greater than or equal to with lhs value.

- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, SelfType>::value)>::type> SelfCrtp< bool, SelfParameters...> operator<= (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> &rhs)

Less than or equal to with lhs value.

- template<typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters>
std::ostream & **operator<<** (std::ostream &lhs, const **StaticVectorizer**< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> &rhs)

Output stream operator.

- template<typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters>
std::istream & **operator>>** (std::istream &lhs, **StaticVectorizer**< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> &rhs)

Input stream operator.

- template<typename SelfType , class SelfPeriod , class SelfClock >
std::ostream & **operator<<** (std::ostream &lhs, const **Timer**< SelfType, SelfPeriod, SelfClock > &rhs)

Output stream operator.

- template<typename SelfType , unsigned int SelfSize>
std::ostream & **operator<<** (std::ostream &lhs, const **Vectorized**< SelfType, SelfSize > &rhs)

Output stream operator.

Locate immutable element from position.

Locates the most refined cell at the provided position and returns an immutable iterator to it.

Template Parameters

Iterator	(Iterator type.)
Types	(Scalar position types.)

Parameters

in	iposs	Real positions along each dimension.
----	-------	--------------------------------------

Returns

Immutable iterator to the element found at the specified position.

- template<typename SelfType , class SelfIndex , class SelfData , unsigned int SelfDimension, class SelfPosition , class SelfExtent , class SelfElement , class SelfContainer >
std::ostream & **operator<<** (std::ostream &lhs, const **SimpleHyperOctree**< SelfType, SelfIndex, SelfData, SelfDimension, SelfPosition, SelfExtent, SelfElement, SelfContainer > &rhs)

Output stream operator.

5.1.1 Function Documentation

- 5.1.1.1 template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, SelfType>::value)>::type> SelfCrtp<bool, SelfParameters...> magrathea::operator!= (const OtherType & lhs, const **StaticVectorizer**< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> &rhs) [inline]

Not equal to with lhs value.

Applies the not equal to operator to each element.

Template Parameters

OtherType	(Other data type.)
SelfKind	(Kind of arguments.)
SelfSize	(Number of elements.)
SelfCrtp	(Derived CRTP class.)
SelfType	(Data type.)
SelfParameters	(List of parameters.)

Parameters

in	<i>lhs</i>	Left-hand side.
in	<i>rhs</i>	Right-hand side.

Returns

Boolean copy.

```
5.1.1.2 template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, SelfType>::value)>::type> SelfCrtp<typename std::common_type<SelfType, OtherType>::type, SelfParameters...> magrathea::operator% ( const OtherType & lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> & rhs ) [inline]
```

Modulo with *lhs* value.

Applies the modulo operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
<i>SelfKind</i>	(Kind of arguments.)
<i>SelfSize</i>	(Number of elements.)
<i>SelfCrtp</i>	(Derived CRTP class.)
<i>SelfType</i>	(Data type.)
<i>SelfParameters</i>	(List of parameters.)

Parameters

in	<i>lhs</i>	Left-hand side.
in	<i>rhs</i>	Right-hand side.

Returns

Common type copy.

```
5.1.1.3 template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, SelfType>::value)>::type> SelfCrtp<typename std::common_type<SelfType, OtherType>::type, SelfParameters...> magrathea::operator& ( const OtherType & lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> & rhs ) [inline]
```

Bitwise AND with *lhs* value.

Applies the bitwise AND operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
<i>SelfKind</i>	(Kind of arguments.)
<i>SelfSize</i>	(Number of elements.)
<i>SelfCrtp</i>	(Derived CRTP class.)
<i>SelfType</i>	(Data type.)
<i>SelfParameters</i>	(List of parameters.)

Parameters

in	<i>lhs</i>	Left-hand side.
in	<i>rhs</i>	Right-hand side.

Returns

Common type copy.

```
5.1.1.4 template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, SelfType>::value)>::type> SelfCrtp<bool, SelfParameters...> magrathea::operator&& ( const OtherType & lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> & rhs ) [inline]
```

Logical AND with *lhs* value.

Applies the logical AND operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
<i>SelfKind</i>	(Kind of arguments.)
<i>SelfSize</i>	(Number of elements.)
<i>SelfCrtp</i>	(Derived CRTP class.)
<i>SelfType</i>	(Data type.)
<i>SelfParameters</i>	(List of parameters.)

Parameters

in	<i>lhs</i>	Left-hand side.
in	<i>rhs</i>	Right-hand side.

Returns

Boolean copy.

```
5.1.1.5 template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, SelfType>::value)>::type> SelfCrtp<typename std::common_type<SelfType, OtherType>::type, SelfParameters...> magrathea::operator* ( const OtherType & lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> & rhs ) [inline]
```

Multiplication with *lhs* value.

Applies the multiplication operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
<i>SelfKind</i>	(Kind of arguments.)
<i>SelfSize</i>	(Number of elements.)
<i>SelfCrtp</i>	(Derived CRTP class.)
<i>SelfType</i>	(Data type.)
<i>SelfParameters</i>	(List of parameters.)

Parameters

in	<i>lhs</i>	Left-hand side.
in	<i>rhs</i>	Right-hand side.

Returns

Common type copy.

```
5.1.1.6 template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, SelfType>::value)>::type> SelfCrtp<typename std::common_type<SelfType, OtherType>::type, SelfParameters...> magrathea::operator+ ( const OtherType & lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> & rhs ) [inline]
```

Addition with *lhs* value.

Applies the addition operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
<i>SelfKind</i>	(Kind of arguments.)
<i>SelfSize</i>	(Number of elements.)
<i>SelfCrtp</i>	(Derived CRTP class.)
<i>SelfType</i>	(Data type.)
<i>SelfParameters</i>	(List of parameters.)

Parameters

in	<i>lhs</i>	Left-hand side.
in	<i>rhs</i>	Right-hand side.

Returns

Common type copy.

```
5.1.1.7 template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, SelfType>::value)>::type> SelfCrtp<typename std::common_type<SelfType, OtherType>::type, SelfParameters...> magrathea::operator- ( const OtherType & lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> & rhs ) [inline]
```

Subtraction with *lhs* value.

Applies the subtraction operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
<i>SelfKind</i>	(Kind of arguments.)
<i>SelfSize</i>	(Number of elements.)
<i>SelfCrtp</i>	(Derived CRTP class.)
<i>SelfType</i>	(Data type.)
<i>SelfParameters</i>	(List of parameters.)

Parameters

in	<i>lhs</i>	Left-hand side.
in	<i>rhs</i>	Right-hand side.

Returns

Common type copy.

```
5.1.1.8 template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, SelfType>::value)>::type> SelfCrtp<typename std::common_type<SelfType, OtherType>::type, SelfParameters...> magrathea::operator/( const OtherType & lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> & rhs ) [inline]
```

Division with *lhs* value.

Applies the division operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
<i>SelfKind</i>	(Kind of arguments.)
<i>SelfSize</i>	(Number of elements.)
<i>SelfCrtp</i>	(Derived CRTP class.)
<i>SelfType</i>	(Data type.)
<i>SelfParameters</i>	(List of parameters.)

Parameters

in	<i>lhs</i>	Left-hand side.
in	<i>rhs</i>	Right-hand side.

Returns

Common type copy.

```
5.1.1.9 template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, SelfType>::value)>::type> SelfCrtp<bool, SelfParameters...> magrathea::operator< ( const OtherType & lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> & rhs ) [inline]
```

Less than with *lhs* value.

Applies the less than operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
<i>SelfKind</i>	(Kind of arguments.)
<i>SelfSize</i>	(Number of elements.)
<i>SelfCrtp</i>	(Derived CRTP class.)
<i>SelfType</i>	(Data type.)
<i>SelfParameters</i>	(List of parameters.)

Parameters

in	<i>lhs</i>	Left-hand side.
in	<i>rhs</i>	Right-hand side.

Returns

Boolean copy.

5.1.1.10 template<typename SelfType , unsigned int SelfSize> std::ostream& magrathea::operator<< (std::ostream & *lhs*, const Vectorized< SelfType, SelfSize > & *rhs*)

[Output](#) stream operator.

Adds each element to the stream using the fill character to separate the elements.

Template Parameters

<i>SelfType</i>	Data type.
<i>SelfSize</i>	Number of elements.

Parameters

in,out	<i>lhs</i>	Left-hand side stream.
in	<i>rhs</i>	Right-hand side container.

Returns

[Output](#) stream.

5.1.1.11 template<typename SelfType , class SelfPeriod , class SelfClock > std::ostream& magrathea::operator<< (std::ostream & *lhs*, const Timer< SelfType, SelfPeriod, SelfClock > & *rhs*)

[Output](#) stream operator.

Prints out the total duration.

Template Parameters

<i>SelfType</i>	(Duration representation type.)
<i>SelfPeriod</i>	(Standard ratio representing the tick period.)
<i>SelfClock</i>	(Internal clock type.)

Parameters

in,out	<i>lhs</i>	Left-hand side stream.
in	<i>rhs</i>	Right-hand side timer.

Returns

[Output](#) stream.

5.1.1.12 std::ostream& magrathea::operator<< (std::ostream & *lhs*, const DataModel & *rhs*)

[Output](#) stream operator.

Prints out the data model.

Parameters

<i>in, out</i>	<i>lhs</i>	Left-hand side stream.
<i>in</i>	<i>rhs</i>	Right-hand side data model.

Returns

[Output](#) stream.

5.1.1.13 template<class *SelfCrtp* , class... *SelfTypes*> std::ostream& magrathea::operator<< (std::ostream & *lhs*, const AbstractSubstance< *SelfCrtp*, *SelfTypes*...> & *rhs*)

[Output](#) stream operator.

Adds each element to the stream.

Template Parameters

<i>SelfCrtp</i>	(Derived CRTP class.)
<i>SelfTypes</i>	(Variadic list of components types.)

Parameters

<i>in, out</i>	<i>lhs</i>	Left-hand side stream.
<i>in</i>	<i>rhs</i>	Right-hand side object.

Returns

[Output](#) stream.

5.1.1.14 template<class *SelfCrtp* , class... *SelfTypes*> std::ostream& magrathea::operator<< (std::ostream & *lhs*, const AbstractAboutObject< *SelfCrtp*, *SelfTypes*...> & *rhs*)

[Output](#) stream operator.

Adds each element to the stream.

Template Parameters

<i>SelfCrtp</i>	(Derived CRTP class.)
<i>SelfTypes</i>	(Variadic list of components types.)

Parameters

<i>in, out</i>	<i>lhs</i>	Left-hand side stream.
<i>in</i>	<i>rhs</i>	Right-hand side object.

Returns

[Output](#) stream.

5.1.1.15 template<class *SelfCrtp* , class *SelfCategory* , class... *SelfTypes*> std::ostream& magrathea::operator<< (std::ostream & *lhs*, const AbstractContents< *SelfCrtp*, *SelfCategory*, *SelfTypes*...> & *rhs*)

[Output](#) stream operator.

Adds each element to the stream.

Template Parameters

<i>SelfCrtp</i>	(Derived CRTP class.)
<i>SelfCategory</i>	(Contents category (Lagrangian, Eulerian, Grid...).)
<i>SelfTypes</i>	(Variadic list of components types.)

Parameters

<i>in, out</i>	<i>lhs</i>	Left-hand side stream.
<i>in</i>	<i>rhs</i>	Right-hand side object.

Returns

[Output](#) stream.

5.1.1.16 std::ostream& magrathea::operator<< (std::ostream & *lhs*, const DataSize & *rhs*)

[Output](#) stream operator.

Prints out the data size in the most convenient binary unit.

Parameters

<i>in, out</i>	<i>lhs</i>	Left-hand side stream.
<i>in</i>	<i>rhs</i>	Right-hand side data size.

Returns

[Output](#) stream.

5.1.1.17 template<class *SelfType* , class *SelfContainer* > std::ostream& magrathea::operator<< (std::ostream & *lhs*, const Evolution< *SelfType*, *SelfContainer* > & *rhs*)

[Output](#) stream operator.

Adds each element to the stream using the `fill()` character as a separator.

Template Parameters

<i>SelfType</i>	(Step type.)
<i>SelfContainer</i>	(Container type.)

Parameters

<i>in,out</i>	<i>lhs</i>	Left-hand side stream.
<i>in</i>	<i>rhs</i>	Right-hand side object.

Returns

[Output](#) stream.

```
5.1.1.18 template<typename SelfType , unsigned int SelfDimension, unsigned int SelfBits> std::ostream&
magrathea::operator<< ( std::ostream & lhs, const SimpleHyperOctreeIndex< SelfType, SelfDimension, SelfBits >
& rhs )
```

[Output](#) stream operator.

Displays the bits of the index.

Template Parameters

<i>SelfType</i>	(Unsigned integer type.)
<i>SelfDimension</i>	(Number of dimensions.)
<i>SelfBits</i>	(Size of the type in bits.)

Parameters

<i>in,out</i>	<i>lhs</i>	Left-hand side stream.
<i>in</i>	<i>rhs</i>	Right-hand side object.

Returns

[Output](#) stream.

```
5.1.1.19 template<typename SelfType , unsigned int SelfSize> std::ostream& magrathea::operator<< ( std::ostream & lhs,
const Constant< SelfType, SelfSize > & rhs )
```

[Output](#) stream operator.

Adds each element to the stream using the `fill()` character as a separator.

Template Parameters

<i>SelfType</i>	(Numerical type of the constant.)
<i>SelfSize</i>	(Size of the set of constants.)

Parameters

<i>in,out</i>	<i>lhs</i>	Left-hand side stream.
<i>in</i>	<i>rhs</i>	Right-hand side constant.

Returns

[Output](#) stream.

```
5.1.1.20 std::ostream& magrathea::operator<< ( std::ostream & lhs, const FileList & rhs )
```

[Output](#) stream operator.

Adds each file name to the stream using the filling character as a separator.

Parameters

<code>in, out</code>	<code>lhs</code>	Left-hand side stream.
<code>in</code>	<code>rhs</code>	Right-hand side file list.

Returns

[Output](#) stream.

```
5.1.1.21 template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename,
SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename
std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType,
SelfType>::value)>::type> SelfCrtp<OtherType, SelfParameters...> magrathea::operator<< ( const OtherType &
lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> & rhs ) [inline]
```

Bitwise left shift with lhs value.

Applies the bitwise left shift operator to each element.

Template Parameters

<code>OtherType</code>	(Other data type.)
<code>SelfKind</code>	(Kind of arguments.)
<code>SelfSize</code>	(Number of elements.)
<code>SelfCrtp</code>	(Derived CRTP class.)
<code>SelfType</code>	(Data type.)
<code>SelfParameters</code>	(List of parameters.)

Parameters

<code>in</code>	<code>lhs</code>	Left-hand side.
<code>in</code>	<code>rhs</code>	Right-hand side.

Returns

Copy.

```
5.1.1.22 template<typename SelfType , class SelfIndex , class SelfData , unsigned int SelfDimension, class SelfPosition ,
class SelfExtent , class SelfElement , class SelfContainer > std::ostream& magrathea::operator<< ( std::ostream &
lhs, const SimpleHyperOctree< SelfType, SelfIndex, SelfData, SelfDimension, SelfPosition, SelfExtent, SelfElement,
SelfContainer > & rhs )
```

[Output](#) stream operator.

Displays the whole structure of the octree.

Template Parameters

<code>SelfType</code>	(Scalar position type.)
<code>SelfIndex</code>	(Index type.)
<code>SelfData</code>	(Data type.)
<code>SelfDimension</code>	(Number of dimensions.)
<code>SelfPosition</code>	(Position of the hyperoctree center.)
<code>SelfExtent</code>	(Extent of the hyperoctree.)
<code>SelfElement</code>	(Underlying element type.)
<code>SelfContainer</code>	(Underlying container type.)

Parameters

<i>in, out</i>	<i>lhs</i>	Left-hand side stream.
<i>in</i>	<i>rhs</i>	Right-hand side object.

Returns

[Output](#) stream.

```
5.1.1.23 template<class SelfCrtp , class SelfScalar , class SelfArray , class SelfTuple > std::ostream& magrathea::operator<<
( std::ostream & lhs, const AbstractStep< SelfCrtp, SelfScalar, SelfArray, SelfTuple > & rhs )
```

[Output](#) stream operator.

Adds each element to the stream.

Template Parameters

<i>SelfCrtp</i>	(Derived CRTP class.)
<i>SelfScalar</i>	(Scalar type of id.)
<i>SelfArray</i>	(Array type of core quantities.)
<i>SelfTuple</i>	(Tuple type of extra quantities.)

Parameters

<i>in, out</i>	<i>lhs</i>	Left-hand side stream.
<i>in</i>	<i>rhs</i>	Right-hand side object.

Returns

[Output](#) stream.

```
5.1.1.24 template<typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename
SelfType , SelfKind... SelfParameters> std::ostream& magrathea::operator<< ( std::ostream & lhs, const
StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> & rhs )
```

[Output](#) stream operator.

Adds each element to the stream using the `fill()` character as a separator.

Template Parameters

<i>SelfKind</i>	(Kind of arguments.)
<i>SelfSize</i>	(Number of elements.)
<i>SelfCrtp</i>	(Derived CRTP class.)
<i>SelfType</i>	(Data type.)
<i>SelfParameters</i>	(List of parameters.)

Parameters

<i>in, out</i>	<i>lhs</i>	Left-hand side stream.
<i>in</i>	<i>rhs</i>	Right-hand side container.

Returns

[Output](#) stream.

```
5.1.1.25 template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename,
SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename
std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType,
SelfType>::value)>::type> SelfCrtp<bool, SelfParameters...> magrathea::operator<= ( const OtherType & lhs,
const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> & rhs ) [inline]
```

Less than or equal to with lhs value.

Applies the less than or equal to operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
<i>SelfKind</i>	(Kind of arguments.)
<i>SelfSize</i>	(Number of elements.)
<i>SelfCrtp</i>	(Derived CRTP class.)
<i>SelfType</i>	(Data type.)
<i>SelfParameters</i>	(List of parameters.)

Parameters

in	<i>lhs</i>	Left-hand side.
in	<i>rhs</i>	Right-hand side.

Returns

Boolean copy.

```
5.1.1.26 template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename,
SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename
std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType,
SelfType>::value)>::type> SelfCrtp<bool, SelfParameters...> magrathea::operator== ( const OtherType & lhs,
const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> & rhs ) [inline]
```

Equal to with lhs value.

Applies the equal to operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
<i>SelfKind</i>	(Kind of arguments.)
<i>SelfSize</i>	(Number of elements.)
<i>SelfCrtp</i>	(Derived CRTP class.)
<i>SelfType</i>	(Data type.)
<i>SelfParameters</i>	(List of parameters.)

Parameters

in	<i>lhs</i>	Left-hand side.
in	<i>rhs</i>	Right-hand side.

Returns

Boolean copy.

```
5.1.1.27 template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename,
SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename
std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType,
SelfType>::value)>::type> SelfCrtp<bool, SelfParameters...> magrathea::operator> ( const OtherType & lhs, const
StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> & rhs ) [inline]
```

Greater than with lhs value.

Applies the greater than operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
<i>SelfKind</i>	(Kind of arguments.)
<i>SelfSize</i>	(Number of elements.)
<i>SelfCrtp</i>	(Derived CRTP class.)
<i>SelfType</i>	(Data type.)
<i>SelfParameters</i>	(List of parameters.)

Parameters

in	<i>lhs</i>	Left-hand side.
in	<i>rhs</i>	Right-hand side.

Returns

Boolean copy.

```
5.1.1.28 template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename,
SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename
std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType,
SelfType>::value)>::type> SelfCrtp<bool, SelfParameters...> magrathea::operator>= ( const OtherType & lhs,
const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> & rhs ) [inline]
```

Greater than or equal to with lhs value.

Applies the greater than or equal to operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
<i>SelfKind</i>	(Kind of arguments.)
<i>SelfSize</i>	(Number of elements.)
<i>SelfCrtp</i>	(Derived CRTP class.)
<i>SelfType</i>	(Data type.)
<i>SelfParameters</i>	(List of parameters.)

Parameters

in	<i>lhs</i>	Left-hand side.
in	<i>rhs</i>	Right-hand side.

Returns

Boolean copy.

```
5.1.1.29 template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename,
SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename
std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType,
SelfType>::value)>::type> SelfCrtp<OtherType, SelfParameters...> magrathea::operator>> ( const OtherType &
lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> & rhs ) [inline]
```

Bitwise right shift with lhs value.

Applies the bitwise right shift operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
<i>SelfKind</i>	(Kind of arguments.)
<i>SelfSize</i>	(Number of elements.)
<i>SelfCrtp</i>	(Derived CRTP class.)
<i>SelfType</i>	(Data type.)
<i>SelfParameters</i>	(List of parameters.)

Parameters

in	<i>lhs</i>	Left-hand side.
in	<i>rhs</i>	Right-hand side.

Returns

Copy.

```
5.1.1.30 template<typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename
SelfType , SelfKind... SelfParameters> std::istream& magrathea::operator>> ( std::istream & lhs, StaticVectorizer<
SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> & rhs )
```

[Input](#) stream operator.

Fills each element from the stream.

Template Parameters

<i>SelfKind</i>	(Kind of arguments.)
<i>SelfSize</i>	(Number of elements.)
<i>SelfCrtp</i>	(Derived CRTP class.)
<i>SelfType</i>	(Data type.)
<i>SelfParameters</i>	(List of parameters.)

Parameters

in, out	<i>lhs</i>	Left-hand side stream.
in	<i>rhs</i>	Right-hand side container.

Returns

[Input](#) stream.

```
5.1.1.31 template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename,
SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename
std::enable_if<!(std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType,
SelfType>::value)>::type> SelfCrtp<typename std::common_type<SelfType, OtherType>::type, SelfParameters...>
magrathea::operator^ ( const OtherType & lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType,
SelfParameters... > & rhs ) [inline]
```

Bitwise XOR with *lhs* value.

Applies the bitwise XOR operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
<i>SelfKind</i>	(Kind of arguments.)
<i>SelfSize</i>	(Number of elements.)
<i>SelfCrtp</i>	(Derived CRTP class.)
<i>SelfType</i>	(Data type.)
<i>SelfParameters</i>	(List of parameters.)

Parameters

in	<i>lhs</i>	Left-hand side.
in	<i>rhs</i>	Right-hand side.

Returns

Common type copy.

```
5.1.1.32 template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename,
SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename
std::enable_if<!(std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType,
SelfType>::value)>::type> SelfCrtp<typename std::common_type<SelfType, OtherType>::type, SelfParameters...>
magrathea::operator| ( const OtherType & lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType,
SelfParameters... > & rhs ) [inline]
```

Bitwise OR with *lhs* value.

Applies the bitwise OR operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
<i>SelfKind</i>	(Kind of arguments.)
<i>SelfSize</i>	(Number of elements.)
<i>SelfCrtp</i>	(Derived CRTP class.)
<i>SelfType</i>	(Data type.)
<i>SelfParameters</i>	(List of parameters.)

Parameters

in	<i>lhs</i>	Left-hand side.
in	<i>rhs</i>	Right-hand side.

Returns

Common type copy.

```
5.1.1.33 template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename,
SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename
std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType,
SelfType>::value)>::type> SelfCrtp<bool, SelfParameters...> magrathea::operator||( const OtherType & lhs, const
StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> & rhs ) [inline]
```

Logical OR with lhs value.

Applies the logical OR operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
<i>SelfKind</i>	(Kind of arguments.)
<i>SelfSize</i>	(Number of elements.)
<i>SelfCrtp</i>	(Derived CRTP class.)
<i>SelfType</i>	(Data type.)
<i>SelfParameters</i>	(List of parameters.)

Parameters

in	<i>lhs</i>	Left-hand side.
in	<i>rhs</i>	Right-hand side.

Returns

Boolean copy.

Chapter 6

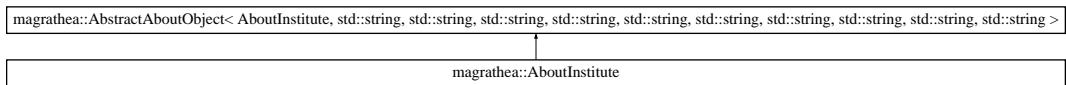
Class Documentation

6.1 magrathea::AboutInstitute Exception Reference

Information about an institution or an organization.

```
#include <aboutinstitute.h>
```

Inheritance diagram for magrathea::AboutInstitute:



Public Member Functions

Lifecycle

- template<class... Misc>
AboutInstitute (Misc &&...misc)
Explicit generic constructor.

Accessors

- std::string **identification** (const std::string &separator=" - ") const
Full identification accessor.
 - std::string **address** (const std::string &separator=", ") const
Address accessor.

Mutators

- **AboutInstitute** & **identification** (const std::string &title, const std::string &sname, const std::string &sex tended)
Full identification mutator.
 - **AboutInstitute** & **address** (const std::string &sstreet, const std::string &szip, const std::string &scity, const std::string &sregion, const std::string &scountry)
Address mutator.

Getters

- const std::string & **title** () const
Title or acronym getter.

- const std::string & **name** () const
Complete name getter.
- const std::string & **extended** () const
Extended name or information getter.
- const std::string & **street** () const
Street number getter.
- const std::string & **zip** () const
Zip code getter.
- const std::string & **city** () const
City getter.
- const std::string & **region** () const
Region getter.
- const std::string & **country** () const
Country getter.
- const std::string & **link** () const
Website or link getter.
- const std::string & **contact** () const
Contact information getter.

Setters

- **AboutInstitute** & **title** (const std::string &value)
Title or acronym setter.
- **AboutInstitute** & **name** (const std::string &value)
Complete name setter.
- **AboutInstitute** & **extended** (const std::string &value)
Extended name or information setter.
- **AboutInstitute** & **street** (const std::string &value)
Street number setter.
- **AboutInstitute** & **zip** (const std::string &value)
Zip code setter.
- **AboutInstitute** & **city** (const std::string &value)
City setter.
- **AboutInstitute** & **region** (const std::string &value)
Region setter.
- **AboutInstitute** & **country** (const std::string &value)
Country setter.
- **AboutInstitute** & **link** (const std::string &value)
Website or link setter.
- **AboutInstitute** & **contact** (const std::string &value)
Contact information setter.

Static Public Member Functions

Predefined

- static const **AboutInstitute** & **luth** ()
Predefined LUTH.
- static const **AboutInstitute** & **sap** ()
Predefined SAP.
- static const **AboutInstitute** & **ipag** ()
Predefined IPAG.
- static const **AboutInstitute** & **obspm** ()
Predefined OBSPM.
- static const **AboutInstitute** & **iap** ()
Predefined IAP.
- static const **AboutInstitute** & **ias** ()
Predefined IAS.

- static const `AboutInstitute & cnrs ()`
Predefined CNRS.
- static const `AboutInstitute & cnes ()`
Predefined CNES.
- static const `AboutInstitute & cea ()`
Predefined CEA.
- static const `AboutInstitute & esa ()`
Predefined ESA.

Test

- static int `example ()`
Example function.

Public Attributes

- using `operator =` = `typedef`

Additional Inherited Members

6.1.1 Detailed Description

Information about an institution or an organization.

Name, link, contact, address... of an institute.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 `template<class... Misc> magrathea::AboutInstitute::AboutInstitute (Misc &&... misc) [inline], [explicit]`

Explicit generic constructor.

Provides a generic interface to all constructors of the base class.

Template Parameters

<code>Misc</code>	(Miscellaneous types.)
-------------------	------------------------

Parameters

<code>in</code>	<code>misc</code>	Miscellaneous arguments.
-----------------	-------------------	--------------------------

6.1.3 Member Function Documentation

6.1.3.1 `std::string magrathea::AboutInstitute::address (const std::string & separator = " ") const [inline]`

Address accessor.

Gets the value of the street, zip, city, region and country properties. If one of the field is empty, the separator is not used.

Parameters

<code>in</code>	<code>separator</code>	String separator.
-----------------	------------------------	-------------------

Returns

Address.

6.1.3.2 AboutInstitute & magrathea::AboutInstitute::address (const std::string & *sstreet*, const std::string & *szip*, const std::string & *scity*, const std::string & *sregion*, const std::string & *scountry*) [inline]

Address mutator.

Sets the value of the street, zip, city, region and country properties.

Parameters

in	<i>sstreet</i>	Street number.
in	<i>szip</i>	Zip code.
in	<i>scity</i>	City.
in	<i>sregion</i>	Region.
in	<i>scountry</i>	Country.

Returns

Self reference.

6.1.3.3 const AboutInstitute & magrathea::AboutInstitute::cea () [inline], [static]

Predefined CEA.

Information on the Commissariat a l'Energie Atomique et aux Energies Alternatives.

Returns

Immutable reference to institute singleton.

6.1.3.4 const std::string & magrathea::AboutInstitute::city () const [inline]

City getter.

Gets the value of the city property.

Returns

City.

6.1.3.5 AboutInstitute & magrathea::AboutInstitute::city (const std::string & *value*) [inline]

City setter.

Sets the value of the city property.

Parameters

in	<i>value</i>	City.
----	--------------	-------

Returns

Self reference.

6.1.3.6 const AboutInstitute & magrathea::AboutInstitute::cnes() [inline], [static]

Predefined CNES.

Information on the Centre National d'Etudes Spatiales.

Returns

Immutable reference to institute singleton.

6.1.3.7 const AboutInstitute & magrathea::AboutInstitute::cnrs() [inline], [static]

Predefined CNRS.

Information on the Centre National de la Recherche Scientifique.

Returns

Immutable reference to institute singleton.

6.1.3.8 const std::string & magrathea::AboutInstitute::contact() const [inline]

Contact information getter.

Gets the value of the contact property.

Returns

Contact.

6.1.3.9 AboutInstitute & magrathea::AboutInstitute::contact(const std::string & value) [inline]

Contact information setter.

Sets the value of the contact property.

Parameters

in	value	Contact.
----	-------	----------

Returns

Self reference.

6.1.3.10 const std::string & magrathea::AboutInstitute::country() const [inline]

Country getter.

Gets the value of the country property.

Returns

Country.

6.1.3.11 AboutInstitute & magrathea::AboutInstitute::country (const std::string & value) [inline]

Country setter.

Sets the value of the country property.

Parameters

in	value	Country.
----	-------	----------

Returns

Self reference.

6.1.3.12 const AboutInstitute & magrathea::AboutInstitute::esa () [inline], [static]

Predefined ESA.

Information on the European Space Agency.

Returns

Immutable reference to institute singleton.

6.1.3.13 int magrathea::AboutInstitute::example () [static]

Example function.

Tests and demonstrates the use of [AboutInstitute](#).

Returns

0 if no error.

6.1.3.14 const std::string & magrathea::AboutInstitute::extended () const [inline]

Extended name or information getter.

Gets the value of the extended property.

Returns

Extended.

6.1.3.15 AboutInstitute & magrathea::AboutInstitute::extended (const std::string & value) [inline]

Extended name or information setter.

Sets the value of the extended property.

Parameters

in	<i>value</i>	Extended.
----	--------------	-----------

Returns

Self reference.

6.1.3.16 const AboutInstitute & magrathea::AboutInstitute::iap() [inline], [static]

Predefined IAP.

Information on the Institut d'Astrophysique de Paris.

Returns

Immutable reference to institute singleton.

6.1.3.17 const AboutInstitute & magrathea::AboutInstitute::ias() [inline], [static]

Predefined IAS.

Information on the Institut d'Astrophysique Spatiale.

Returns

Immutable reference to institute singleton.

6.1.3.18 std::string magrathea::AboutInstitute::identification(const std::string & separator = " - ") const [inline]

Full identification accessor.

Gets the value of the title, the name and the extended name properties. If one of the field is empty, the separator is not used.

Parameters

in	<i>separator</i>	String separator.
----	------------------	-------------------

Returns

Identification.

6.1.3.19 AboutInstitute & magrathea::AboutInstitute::identification(const std::string & stitle, const std::string & sname, const std::string & sextended) [inline]

Full identification mutator.

Sets the value of the title, the name and the extended name properties.

Parameters

in	<i>stitle</i>	Title or acronym.
in	<i>sname</i>	Complete name.
in	<i>sextended</i>	Extended name or information.

Returns

Self reference.

6.1.3.20 const AboutInstitute & magrathea::AboutInstitute::ipag() [inline], [static]

Predefined IPAG.

Information on the Institut de Planetologie et d'Astrophysique de Grenoble

Returns

Immutable reference to institute singleton.

6.1.3.21 const std::string & magrathea::AboutInstitute::link() const [inline]

Website or link getter.

Gets the value of the link property.

Returns

Link.

6.1.3.22 AboutInstitute & magrathea::AboutInstitute::link(const std::string & value) [inline]

Website or link setter.

Sets the value of the link property.

Parameters

in	value	Link.
----	-------	-------

Returns

Self reference.

6.1.3.23 const AboutInstitute & magrathea::AboutInstitute::luth() [inline], [static]

Predefined LUTH.

Information on the Laboratoire Univers et Theories.

Returns

Immutable reference to institute singleton.

6.1.3.24 const std::string & magrathea::AboutInstitute::name() const [inline]

Complete name getter.

Gets the value of the name property.

Returns

Name.

6.1.3.25 AboutInstitute & magrathea::AboutInstitute::name (const std::string & value) [inline]

Complete name setter.

Sets the value of the name property.

Parameters

in	value	Name.
----	-------	-------

Returns

Self reference.

6.1.3.26 const AboutInstitute & magrathea::AboutInstitute::obspm () [inline], [static]

Predefined OBSPM.

Information on the Observatoire de Paris.

Returns

Immutable reference to institute singleton.

6.1.3.27 const std::string & magrathea::AboutInstitute::region () const [inline]

Region getter.

Gets the value of the region property.

Returns

Region.

6.1.3.28 AboutInstitute & magrathea::AboutInstitute::region (const std::string & value) [inline]

Region setter.

Sets the value of the region property.

Parameters

in	value	Region.
----	-------	---------

Returns

Self reference.

6.1.3.29 const AboutInstitute & magrathea::AboutInstitute::sap () [inline], [static]

Predefined SAp.

Information on the Service d'Astrophysique du CEA Saclay.

Returns

Immutable reference to institute singleton.

6.1.3.30 const std::string & magrathea::AboutInstitute::street() const [inline]

Street number getter.

Gets the value of the street property.

Returns

Street.

6.1.3.31 AboutInstitute & magrathea::AboutInstitute::street(const std::string & value) [inline]

Street number setter.

Sets the value of the street property.

Parameters

in	value	Street.
----	-------	---------

Returns

Self reference.

6.1.3.32 const std::string & magrathea::AboutInstitute::title() const [inline]

Title or acronym getter.

Gets the value of the title property.

Returns

Title.

6.1.3.33 AboutInstitute & magrathea::AboutInstitute::title(const std::string & value) [inline]

Title or acronym setter.

Sets the value of the title property.

Parameters

in	value	Title.
----	-------	--------

Returns

Self reference.

6.1.3.34 const std::string & magrathea::AboutInstitute::zip() const [inline]

Zip code getter.

Gets the value of the zip property.

Returns

Zip.

6.1.3.35 AboutInstitute & magrathea::AboutInstitute::zip (const std::string & value) [inline]

Zip code setter.

Sets the value of the zip property.

Parameters

in	value	Zip.
----	-------	------

Returns

Self reference.

6.1.4 Member Data Documentation

6.1.4.1 using magrathea::AboutInstitute::operator =

The documentation for this exception was generated from the following file:

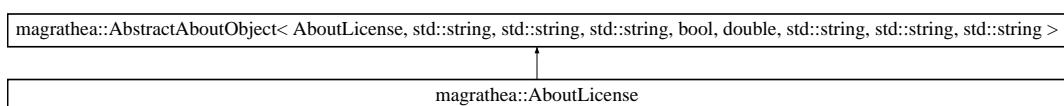
- /data/home/mbreton/magrathea_pathfinder/src/magrathea/aboutinstitute.h

6.2 magrathea::AboutLicense Exception Reference

Information about the license of a code.

```
#include <aboutlicense.h>
```

Inheritance diagram for magrathea::AboutLicense:



Public Member Functions

Lifecycle

- template<class... Misc>
AboutLicense (Misc &&...misc)
Explicit generic constructor.

Accessors

- std::string **text** (const std::string &program, const std::string &date="", const std::string &owner="", const std::string &contact="", const std::string &organization="", const std::string &purpose="") const
License text of the program accessor.

Getters

- const std::string & **title** () const
License title, name or acronym getter.
- const std::string & **name** () const
Complete name getter.
- const std::string & **link** () const

- **bool proprietary () const**
Proprietary or free/open-source flag getter.
- **double version () const**
Version number getter.
- **const std::string & description () const**
Short description or summary of the license getter.
- **const std::string & text () const**
License text of the program getter.
- **const std::string & reference () const**
Complete license reference getter.

Setters

- **AboutLicense & title (const std::string &value)**
License title, name or acronym setter.
- **AboutLicense & name (const std::string &value)**
Complete name setter.
- **AboutLicense & link (const std::string &value)**
Web link setter.
- **AboutLicense & proprietary (const bool value)**
Proprietary or free/open-source flag setter.
- **AboutLicense & version (const double value)**
Version number setter.
- **AboutLicense & description (const std::string &value)**
Short description or summary of the license setter.
- **AboutLicense & text (const std::string &value)**
License text of the program setter.
- **AboutLicense & reference (const std::string &value)**
Complete license reference setter.

Static Public Member Functions

Predefined

- **static const AboutLicense & gplv3 ()**
Predefined GPLv3 license.
- **static const AboutLicense & lgplv3 ()**
Predefined LGPLv3 license.
- **static const AboutLicense & bsd ()**
Predefined BSD license.
- **static const AboutLicense & modifiedbsd ()**
Predefined modified BSD license.
- **static const AboutLicense & freebsd ()**
Predefined FreeBSD license.
- **static const AboutLicense & cecill ()**
Predefined CeCILL license.
- **static const AboutLicense & cecillb ()**
Predefined CeCILL-B license.
- **static const AboutLicense & cecillc ()**
Predefined CeCILL-C license.
- **static const AboutLicense & ccbyv3 ()**
Predefined CC-BY v3 license.
- **static const AboutLicense & ccbysav3 ()**
Predefined CC-BY-SA v3 license.
- **static const AboutLicense & ccbyndv3 ()**
Predefined CC-BY-ND v3 license.
- **static const AboutLicense & ccbyncv3 ()**

- Predefined CC-BY-NC v3 license.
• static const [AboutLicense & ccbyncsv3 \(\)](#)
- Predefined CC-BY-NC-SA v3 license.
• static const [AboutLicense & ccbyncndv3 \(\)](#)
- Predefined CC-BY-NC-ND v3 license.
• static const [AboutLicense & cc0v1 \(\)](#)

Predefined CC0 v1 license.

Test

- static int [example \(\)](#)
Example function.

Public Attributes

- using [operator =](#) = [typedef](#)

Additional Inherited Members

6.2.1 Detailed Description

Information about the license of a code.

Name, link, text and documentation on the license of a code or a library.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 `template<class... Misc> magrathea::AboutLicense::AboutLicense (Misc &&... misc) [inline], [explicit]`

Explicit generic constructor.

Provides a generic interface to all constructors of the base class.

Template Parameters

<code>Misc</code>	(Miscellaneous types.)
-------------------	------------------------

Parameters

<code>in</code>	<code>misc</code>	Miscellaneous arguments.
-----------------	-------------------	--------------------------

6.2.3 Member Function Documentation

6.2.3.1 `const AboutLicense & magrathea::AboutLicense::bsd () [inline], [static]`

Predefined BSD license.

Original BSD license.

Returns

Immutable reference to license singleton.

6.2.3.2 const AboutLicense & magrathea::AboutLicense::cc0v1() [inline], [static]

Predefined CC0 v1 license.

Creative Commons CC0 1.0 Universal Public Domain Dedication.

Returns

Immutable reference to license singleton.

6.2.3.3 const AboutLicense & magrathea::AboutLicense::ccbyncndv3() [inline], [static]

Predefined CC-BY-NC-ND v3 license.

Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported (CC BY-NC-ND 3.0).

Returns

Immutable reference to license singleton.

6.2.3.4 const AboutLicense & magrathea::AboutLicense::ccbyncsav3() [inline], [static]

Predefined CC-BY-NC-SA v3 license.

Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported (CC BY-NC-SA 3.0).

Returns

Immutable reference to license singleton.

6.2.3.5 const AboutLicense & magrathea::AboutLicense::ccbyncv3() [inline], [static]

Predefined CC-BY-NC v3 license.

Creative Commons Attribution-NonCommercial 3.0 Unported (CC BY-NC 3.0).

Returns

Immutable reference to license singleton.

6.2.3.6 const AboutLicense & magrathea::AboutLicense::ccbnyndv3() [inline], [static]

Predefined CC-BY-ND v3 license.

Creative Commons Attribution-NoDerivs 3.0 Unported (CC BY-ND 3.0).

Returns

Immutable reference to license singleton.

6.2.3.7 const AboutLicense & magrathea::AboutLicense::ccbysav3() [inline], [static]

Predefined CC-BY-SA v3 license.

Creative Commons Attribution-ShareAlike 3.0 Unported (CC BY-SA 3.0).

Returns

Immutable reference to license singleton.

6.2.3.8 const AboutLicense & magrathea::AboutLicense::ccbyv3() [inline], [static]

Predefined CC-BY v3 license.

Creative Commons Attribution 3.0 Unported (CC BY 3.0).

Returns

Immutable reference to license singleton.

6.2.3.9 const AboutLicense & magrathea::AboutLicense::cecill() [inline], [static]

Predefined CeCILL license.

CeCILL license.

Returns

Immutable reference to license singleton.

6.2.3.10 const AboutLicense & magrathea::AboutLicense::cecillb() [inline], [static]

Predefined CeCILL-B license.

CeCILL-B license.

Returns

Immutable reference to license singleton.

6.2.3.11 const AboutLicense & magrathea::AboutLicense::cecillc() [inline], [static]

Predefined CeCILL-C license.

CeCILL-C license.

Returns

Immutable reference to license singleton.

6.2.3.12 const std::string & magrathea::AboutLicense::description() const [inline]

Short description or summary of the license getter.

Gets the value of the description property.

Returns

Description.

6.2.3.13 AboutLicense & magrathea::AboutLicense::description(const std::string & value) [inline]

Short description or summary of the license setter.

Sets the value of the description property.

Parameters

in	value	Description.
----	-------	--------------

Returns

Self reference.

6.2.3.14 int magrathea::AboutLicense::example() [static]

Example function.

Tests and demonstrates the use of [AboutLicense](#).

Returns

0 if no error.

6.2.3.15 const AboutLicense & magrathea::AboutLicense::freebsd() [inline], [static]

Predefined FreeBSD license.

FreeBSD license.

Returns

Immutable reference to license singleton.

6.2.3.16 const AboutLicense & magrathea::AboutLicense::gplv3() [inline], [static]

Predefined GPLv3 license.

GNU General Public License Version 3.0.

Returns

Immutable reference to license singleton.

6.2.3.17 const AboutLicense & magrathea::AboutLicense::lgplv3() [inline], [static]

Predefined LGPLv3 license.

GNU Lesser General Public License Version 3.0.

Returns

Immutable reference to license singleton.

6.2.3.18 const std::string & magrathea::AboutLicense::link() const [inline]

Web link getter.

Gets the value of the link property.

Returns

Link.

6.2.3.19 AboutLicense & magrathea::AboutLicense::link (const std::string & value) [inline]

Web link setter.

Sets the value of the link property.

Parameters

in	value	Link.
----	-------	-------

Returns

Self reference.

6.2.3.20 const AboutLicense & magrathea::AboutLicense::modifiedbsd() [inline], [static]

Predefined modified BSD license.

Modified BSD license.

Returns

Immutable reference to license singleton.

6.2.3.21 const std::string & magrathea::AboutLicense::name() const [inline]

Complete name getter.

Gets the value of the name property.

Returns

Name.

6.2.3.22 AboutLicense & magrathea::AboutLicense::name(const std::string & value) [inline]

Complete name setter.

Sets the value of the name property.

Parameters

in	value	Name.
----	-------	-------

Returns

Self reference.

6.2.3.23 bool magrathea::AboutLicense::proprietary() const [inline]

Proprietary or free/open-source flag getter.

Gets the value of the proprietary property.

Returns

Proprietary.

6.2.3.24 AboutLicense & magrathea::AboutLicense::proprietary (const bool *value*) [inline]

Proprietary or free/open-source flag setter.

Sets the value of the proprietary property.

Parameters

in	<i>value</i>	Proprietary.
----	--------------	--------------

Returns

Self reference.

6.2.3.25 const std::string & magrathea::AboutLicense::reference () const [inline]

Complete license reference getter.

Gets the value of the reference property.

Returns

Reference.

6.2.3.26 AboutLicense & magrathea::AboutLicense::reference (const std::string & *value*) [inline]

Complete license reference setter.

Sets the value of the reference property.

Parameters

in	<i>value</i>	Reference.
----	--------------	------------

Returns

Self reference.

6.2.3.27 std::string magrathea::AboutLicense::text (const std::string & *program*, const std::string & *date* = "", const std::string & *owner* = "", const std::string & *contact* = "", const std::string & *organization* = "", const std::string & *purpose* = "") const [inline]

License text of the program accessor.

Gets the value of the text property with correctly replaced keywords.

Returns

Text.

6.2.3.28 const std::string & magrathea::AboutLicense::text () const [inline]

License text of the program getter.

Gets the value of the text property.

Returns

Text.

6.2.3.29 AboutLicense & magrathea::AboutLicense::text (const std::string & value) [inline]

License text of the program setter.

Sets the value of the text property.

Parameters

in	value	Text.
----	-------	-------

Returns

Self reference.

6.2.3.30 const std::string & magrathea::AboutLicense::title () const [inline]

License title, name or acronym getter.

Gets the value of the title property.

Returns

Title.

6.2.3.31 AboutLicense & magrathea::AboutLicense::title (const std::string & value) [inline]

License title, name or acronym setter.

Sets the value of the title property.

Parameters

in	value	Title.
----	-------	--------

Returns

Self reference.

6.2.3.32 double magrathea::AboutLicense::version () const [inline]

Version number getter.

Gets the value of the version property.

Returns

Version.

6.2.3.33 AboutLicense & magrathea::AboutLicense::version (const double value) [inline]

Version number setter.

Sets the value of the version property.

Parameters

in	value	Version.
----	-------	----------

Returns

Self reference.

6.2.4 Member Data Documentation

6.2.4.1 using magrathea::AboutLicense::operator =

The documentation for this exception was generated from the following file:

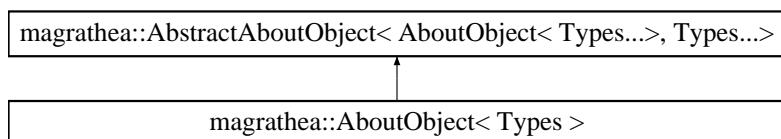
- /data/home/mbreton/magrathea_pathfinder/src/magrathea/aboutlicense.h

6.3 magrathea::AboutObject< Types > Exception Template Reference

Basic about object with information on something.

```
#include <aboutobject.h>
```

Inheritance diagram for magrathea::AboutObject< Types >:



Public Member Functions

Lifecycle

- template<class... Misc>
[AboutObject](#) (Misc &&...misc)
Explicit generic constructor.

Static Public Member Functions

Test

- static int [example](#) ()
Example function.

Public Attributes

- using [operator](#) = typedef

Additional Inherited Members

6.3.1 Detailed Description

```
template<class... Types>exception magrathea::AboutObject< Types >
```

Basic about object with information on something.

This class is the direct derivation of [AbstractAboutObject](#). It provides the most basic and generic contents object without adding new functionalities to the abstract class. It can be used in most cases as a generic container of groups of information on something.

Template Parameters

Types	Variadic list of components types.
-------	------------------------------------

6.3.2 Constructor & Destructor Documentation

```
6.3.2.1 template<class... Types> template<class... Misc> magrathea::AboutObject< Types >::AboutObject( Misc  
    &&... misc ) [inline], [explicit]
```

Explicit generic constructor.

Provides a generic interface to all constructors of the base class.

Template Parameters

Misc	(Miscellaneous types.)
------	------------------------

Parameters

in	misc	Miscellaneous arguments.
----	------	--------------------------

6.3.3 Member Function Documentation

```
6.3.3.1 template<class... Types> int magrathea::AboutObject< Types >::example( ) [static]
```

Example function.

Tests and demonstrates the use of [AboutObject](#).

Returns

0 if no error.

6.3.4 Member Data Documentation

```
6.3.4.1 template<class... Types> using magrathea::AboutObject< Types >::operator =
```

The documentation for this exception was generated from the following file:

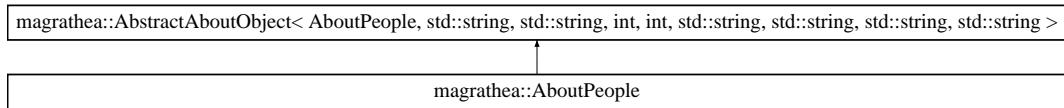
- /data/home/mbreton/magrathea_pathfinder/src/magrathea/aboutobject.h

6.4 magrathea::AboutPeople Exception Reference

Information about a developer, an author, or a contributor.

```
#include <aboutpeople.h>
```

Inheritance diagram for magrathea::AboutPeople:



Public Member Functions

Lifecycle

- template<class... Misc>
[AboutPeople](#) (Misc &&...misc)
Explicit generic constructor.

Accessors

- std::string [name](#) (const std::string &separator=" ") const
Full name accessor.
- std::string [years](#) (const std::string &separator="-") const
Year range accessor.
- std::string [mails](#) (const std::string &separator=" - ") const
Mails accessor.

Mutators

- [AboutPeople & name](#) (const std::string &fname, const std::string &lname)
Full name mutator.
- [AboutPeople & years](#) (const int fyear, const int lyear)
Year range mutator.
- [AboutPeople & mails](#) (const std::string &fmail, const std::string &lmail)
Mails mutator.

Getters

- const std::string & [first](#) () const
First name getter.
- const std::string & [last](#) () const
Last name getter.
- int [begin](#) () const
First year of contribution getter.
- int [end](#) () const
Last year of contribution getter.
- const std::string & [mail](#) () const
E-mail getter.
- const std::string & [altmail](#) () const
Alternative e-mail getter.
- const std::string & [link](#) () const
Web link getter.
- const std::string & [contact](#) () const
Additional contact information getter.

Setters

- [AboutPeople & first](#) (const std::string &value)

- *First name setter.*
 • `AboutPeople & last` (const std::string &value)
Last name setter.
- `AboutPeople & begin` (const int value)
First year of contribution setter.
- `AboutPeople & end` (const int value)
Last year of contribution setter.
- `AboutPeople & mail` (const std::string &value)
E-mail setter.
- `AboutPeople & altmail` (const std::string &value)
Alternative e-mail setter.
- `AboutPeople & link` (const std::string &value)
Web link setter.
- `AboutPeople & contact` (const std::string &value)
Additional contact information setter.

Static Public Member Functions

Predefined

- static const `AboutPeople & vreverdy` ()
Predefined Vincent Reverdy.

Test

- static int `example` ()
Example function.

Public Attributes

- using `operator =` = `typedef`

Additional Inherited Members

6.4.1 Detailed Description

Information about a developer, an author, or a contributor.

Name, status, contact, link, dates of contributions... for authors and contributors.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 template<class... Misc> magrathea::AboutPeople::AboutPeople (`Misc &&... misc`) [inline], [explicit]

Explicit generic constructor.

Provides a generic interface to all constructors of the base class.

Template Parameters

<code>Misc</code>	(Miscellaneous types.)
-------------------	------------------------

Parameters

<code>in</code>	<code>misc</code>	Miscellaneous arguments.
-----------------	-------------------	--------------------------

6.4.3 Member Function Documentation

6.4.3.1 const std::string & magrathea::AboutPeople::altnail () const [inline]

Alternative e-mail getter.

Gets the value of the altnail property.

Returns

Altnail.

6.4.3.2 AboutPeople & magrathea::AboutPeople::altnail (const std::string & value) [inline]

Alternative e-mail setter.

Sets the value of the altnail property.

Parameters

in	value	Altnail.
----	-------	----------

Returns

Self reference.

6.4.3.3 int magrathea::AboutPeople::begin () const [inline]

First year of contribution getter.

Gets the value of the begin property.

Returns

Begin.

6.4.3.4 AboutPeople & magrathea::AboutPeople::begin (const int value) [inline]

First year of contribution setter.

Sets the value of the begin property.

Parameters

in	value	Begin.
----	-------	--------

Returns

Self reference.

6.4.3.5 const std::string & magrathea::AboutPeople::contact () const [inline]

Additional contact information getter.

Gets the value of the contact property.

Returns

Contact.

6.4.3.6 AboutPeople & magrathea::AboutPeople::contact (const std::string & value) [inline]

Additional contact information setter.

Sets the value of the contact property.

Parameters

in	value	Contact.
----	-------	----------

Returns

Self reference.

6.4.3.7 int magrathea::AboutPeople::end () const [inline]

Last year of contribution getter.

Gets the value of the end property.

Returns

End.

6.4.3.8 AboutPeople & magrathea::AboutPeople::end (const int value) [inline]

Last year of contribution setter.

Sets the value of the end property.

Parameters

in	value	End.
----	-------	------

Returns

Self reference.

6.4.3.9 int magrathea::AboutPeople::example () [static]

Example function.

Tests and demonstrates the use of [AboutPeople](#).

Returns

0 if no error.

6.4.3.10 const std::string & magrathea::AboutPeople::first () const [inline]

First name getter.

Gets the value of the first property.

Returns

First.

6.4.3.11 AboutPeople & magrathea::AboutPeople::first (const std::string & value) [inline]

Last name getter.

Gets the value of the first property.

Parameters

in	value	First.
----	-------	--------

Returns

Self reference.

6.4.3.12 const std::string & magrathea::AboutPeople::last () const [inline]

Last name getter.

Gets the value of the last property.

Returns

Last.

6.4.3.13 AboutPeople & magrathea::AboutPeople::last (const std::string & value) [inline]

Last name setter.

Sets the value of the last property.

Parameters

in	value	Last.
----	-------	-------

Returns

Self reference.

6.4.3.14 const std::string & magrathea::AboutPeople::link () const [inline]

Web link getter.

Gets the value of the link property.

Returns

Link.

6.4.3.15 AboutPeople & magrathea::AboutPeople::link (const std::string & value) [inline]

Web link setter.

Sets the value of the link property.

Parameters

in	<i>value</i>	Link.
----	--------------	-------

Returns

Self reference.

6.4.3.16 const std::string & magrathea::AboutPeople::mail() const [inline]

E-mail getter.

Gets the value of the mail property.

Returns

Mail.

6.4.3.17 AboutPeople & magrathea::AboutPeople::mail(const std::string & value) [inline]

E-mail setter.

Sets the value of the mail property.

Parameters

in	<i>value</i>	Mail.
----	--------------	-------

Returns

Self reference.

6.4.3.18 std::string magrathea::AboutPeople::mails(const std::string & separator = " - ") const [inline]

Mails accessor.

Gets the value of the main and alternative mails properties. If one of the mails is empty, the separator is not used.

Parameters

in	<i>separator</i>	String separator.
----	------------------	-------------------

Returns

Mails.

6.4.3.19 AboutPeople & magrathea::AboutPeople::mails(const std::string & fmail, const std::string & lmail) [inline]

Mails mutator.

Sets the value of the main and alternative mails properties.

Parameters

in	<i>fmail</i>	E-mail.
in	<i>lmail</i>	Alternative e-mail.

Returns

Mails.

6.4.3.20 std::string magrathea::AboutPeople::name (const std::string & separator = " ") const [inline]

Full name accessor.

Gets the value of the first and last name properties. If one of the first or last name is empty, the separator is not used.

Parameters

in	<i>separator</i>	String separator.
----	------------------	-------------------

Returns

Name.

6.4.3.21 AboutPeople & magrathea::AboutPeople::name (const std::string & fname, const std::string & lname) [inline]

Full name mutator.

Sets the value of the first and last name properties.

Parameters

in	<i>fname</i>	First name.
in	<i>lname</i>	Last name.

Returns

Self reference.

6.4.3.22 const AboutPeople & magrathea::AboutPeople::vreverdy () [inline], [static]

Predefined Vincent Reverdy.

Vincent Reverdy details.

Returns

Immutable reference to people singleton.

6.4.3.23 std::string magrathea::AboutPeople::years (const std::string & separator = "-") const [inline]

Year range accessor.

Gets the value of the begin and end year properties. If the end year is lower than the begin year, it is not displayed.

Parameters

in	<i>separator</i>	String separator.
----	------------------	-------------------

Returns

Years.

6.4.3.24 **AboutPeople & magrathea::AboutPeople::years** (const int *fyear*, const int *lyear*) [inline]

Year range mutator.

Sets the value of the begin and end year properties.

Parameters

in	<i>fyear</i>	First year of contribution.
in	<i>lyear</i>	Last year of contribution.

Returns

Self reference.

6.4.4 Member Data Documentation

6.4.4.1 using `magrathea::AboutPeople::operator =`

The documentation for this exception was generated from the following file:

- /data/home/mbreton/magrathea_pathfinder/src/magrathea/aboutpeople.h

6.5 `magrathea::AbstractAboutObject< Crtp, Types >` Class Template Reference

Tuple abstraction of generic about object.

```
#include <abstractaboutobject.h>
```

Public Member Functions

Lifecycle

- **AbstractAboutObject ()**
Implicit empty constructor.
 - template<class OtherCrtp , class... OtherTypes>
AbstractAboutObject (const **AbstractAboutObject**< OtherCrtp, OtherTypes...> &source)
Explicit conversion constructor.
 - template<class... OtherTypes>
AbstractAboutObject (const std::tuple< OtherTypes...> &source)
Explicit data constructor.
 - template<class... OtherTypes, class = typename std::enable_if<(sizeof...(OtherTypes) != 0) && (std::is_constructible<typename std::tuple_element<0, typename std::conditional<sizeof...(Types) != 0, std::tuple<Types...>, std::tuple<std::true_type> >::type>-::type, typename std::tuple_element<0, typename std::conditional<sizeof...(OtherTypes) != 0, std::tuple<OtherTypes...>, std::tuple<std::true_type> >::type>::type>::value>>::type>
AbstractAboutObject (const OtherTypes &...source)
Explicit components constructor.

Operators

- Crtp & `operator=` (const `AbstractAboutObject< Crtp, Types...>` &rhs)

- template<class OtherCrtp , class... OtherTypes>

Crtp & **operator=** (const **AbstractAboutObject**< OtherCrtp, OtherTypes...> &rhs)

Conversion assignment operator.
- template<class... OtherTypes>

Crtp & **operator=** (const std::tuple< OtherTypes...> &rhs)

Data assignment operator.
- template<class OtherCrtp , class... OtherTypes>

bool **operator==** (const **AbstractAboutObject**< OtherCrtp, OtherTypes...> &rhs) const

Equal to.
- template<class OtherCrtp , class... OtherTypes>

bool **operator!=** (const **AbstractAboutObject**< OtherCrtp, OtherTypes...> &rhs) const

Not equal to.

Assignment

- Crtp & **assign** ()

Empty assignment.
- Crtp & **assign** (const **AbstractAboutObject**< Crtp, Types...> &source)

Copy assignment.
- template<class OtherCrtp , class... OtherTypes>

Crtp & **assign** (const **AbstractAboutObject**< OtherCrtp, OtherTypes...> &source)

Conversion assignment.
- template<class... OtherTypes>

Crtp & **assign** (const std::tuple< OtherTypes...> &source)

Data assignment.
- template<class... OtherTypes, class = typename std::enable_if<(sizeof...(OtherTypes) != 0) && (std::is_constructible<typename std::tuple_element<0, typename std::conditional<sizeof...(Types) != 0, std::tuple<Types...>, std::tuple<std::true_type> >::type>-::type, typename std::tuple_element<0, typename std::conditional<sizeof...(OtherTypes) != 0, std::tuple<OtherTypes...>, std::tuple<std::true_type> >::type>::value>::type>

Crtp & **assign** (const OtherTypes &...source)

Components assignment.

Management

- Crtp & **nullify** ()

Nullify.
- Crtp **copy** () const

Copy.
- template<class OtherCrtp = Crtp, class = typename std::enable_if<std::is_constructible<OtherCrtp, Crtp>::value>::type>

OtherCrtp **cast** () const

Cast.

Data

- template<class... Dummy, class Type = typename std::conditional<sizeof...(Dummy) == 0, std::tuple<Types...>, void>::type, class = typename std::enable_if<sizeof...(Dummy) == 0>::type, class = typename std::enable_if<std::is_convertible<Type, typename std::conditional<sizeof...(Dummy) == 0, std::tuple<Types...>, void>::type>::value>::type>

Type & data (Dummy...)

Unified data access.
- template<class... Dummy, class Type = typename std::conditional<sizeof...(Dummy) == 0, std::tuple<Types...>, void>::type, class = typename std::enable_if<sizeof...(Dummy) == 0>::type, class = typename std::enable_if<std::is_convertible<Type, typename std::conditional<sizeof...(Dummy) == 0, std::tuple<Types...>, void>::type>::value>::type>

const Type & **data** (Dummy...) const

Unified data getter.
- template<class Type , class = typename std::enable_if<std::is_convertible<Type, typename std::conditional<!std::is_void<Type>-::value, std::tuple<Types...>, void>::type>::value>::type>

Crtp & **data** (const Type &value)

Unified data setter.

Protected Member Functions

Protected lifecycle

- `~AbstractAboutObject ()`

Protected destructor.

6.5.1 Detailed Description

```
template<class Crtp, class... Types>class magrathea::AbstractAboutObject< Crtp, Types >
```

Tuple abstraction of generic about object.

This class is an abstraction of an about object with common functions, like assignment, copy, getters and setters. An about object is basically a standard structure with additional features in order to be able to group information about a software, its authors, its license and other information.

Template Parameters

<i>Crtp</i>	Derived CRTP class.
<i>Types</i>	Variadic list of components types.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 `template<class Crtp , class... Types> magrathea::AbstractAboutObject< Crtp, Types >::~AbstractAboutObject() [inline], [protected], [default]`

Protected destructor.

Avoids direct instantiation of the class, and only allows it through its derived children.

6.5.2.2 `template<class Crtp , class... Types> magrathea::AbstractAboutObject< Crtp, Types >::AbstractAboutObject() [inline]`

Implicit empty constructor.

Provides an implicit construction of an object initialized to its default value.

6.5.2.3 `template<class Crtp , class... Types> template<class OtherCrtp , class... OtherTypes> magrathea::AbstractAboutObject< Crtp, Types >::AbstractAboutObject (const AbstractAboutObject< OtherCrtp, OtherTypes...> & source) [inline], [explicit]`

Explicit conversion constructor.

Provides an explicit construction from another type of object.

Template Parameters

<i>OtherCrtp</i>	(Other derived CRTP class.)
<i>OtherTypes</i>	(Other variadic list of components types.)

Parameters

<i>in</i>	<i>source</i>	Source of the copy.
-----------	---------------	---------------------

6.5.2.4 template<class Crtp , class... Types> template<class... OtherTypes> **magrathea::AbstractAboutObject< Crtp, Types >::AbstractAboutObject** (const std::tuple< OtherTypes...> & source) [inline], [explicit]

Explicit data constructor.

Provides an explicit construction from data.

Template Parameters

<i>OtherTypes</i>	(Other variadic list of object property types.)
-------------------	---

Parameters

in	<i>source</i>	Source of the copy.
----	---------------	---------------------

6.5.2.5 template<class Crtp , class... Types> template<class... OtherTypes, class > **magrathea::AbstractAboutObject< Crtp, Types >::AbstractAboutObject** (const OtherTypes &... source) [inline], [explicit]

Explicit components constructor.

Provides an explicit construction from components.

Template Parameters

<i>OtherTypes</i>	(Other variadic list of object property types.)
-------------------	---

Parameters

in	<i>source</i>	Source of the copy.
----	---------------	---------------------

6.5.3 Member Function Documentation

6.5.3.1 template<class Crtp , class... Types> Crtp & **magrathea::AbstractAboutObject< Crtp, Types >::assign** () [inline]

Empty assignment.

Assigns contents from an object initialized to its default value.

Returns

Self reference.

6.5.3.2 template<class Crtp, class... Types> Crtp & **magrathea::AbstractAboutObject< Crtp, Types >::assign** (const AbstractAboutObject< Crtp, Types...> & source) [inline]

Copy assignment.

Assigns contents from the same type of object.

Parameters

in	<i>source</i>	Source of the copy.
----	---------------	---------------------

Returns

Self reference.

6.5.3.3 template<class Crtp , class... Types> template<class OtherCrtp , class... OtherTypes> Crtp & magrathea::AbstractAboutObject< Crtp, Types >::assign (const AbstractAboutObject< OtherCrtp, OtherTypes...> & source) [inline]

Conversion assignment.

Assigns contents from another type of object.

Template Parameters

<i>OtherCrtp</i>	(Other derived CRTP class.)
<i>OtherTypes</i>	(Other variadic list of components types.)

Parameters

<i>in</i>	<i>source</i>	Source of the copy.
-----------	---------------	---------------------

Returns

Self reference.

6.5.3.4 template<class Crtp , class... Types> template<class... OtherTypes> Crtp & magrathea::AbstractAboutObject< Crtp, Types >::assign (const std::tuple< OtherTypes...> & source) [inline]

Data assignment.

Assigns contents from data.

Template Parameters

<i>OtherTypes</i>	(Other variadic list of object property types.)
-------------------	---

Parameters

<i>in</i>	<i>source</i>	Source of the copy.
-----------	---------------	---------------------

Returns

Self reference.

6.5.3.5 template<class Crtp , class... Types> template<class... OtherTypes, class > Crtp & magrathea::AbstractAboutObject< Crtp, Types >::assign (const OtherTypes &... source) [inline]

Components assignment.

Assigns contents from components.

Template Parameters

<i>OtherTypes</i>	(Other variadic list of object property types.)
-------------------	---

Parameters

in	source	Source of the copy.
----	--------	---------------------

Returns

Self reference.

**6.5.3.6 template<class Crtp , class... Types> template<class OtherCrtp , class > OtherCrtp
magrathea::AbstractAboutObject< Crtp, Types >::cast() const [inline]**

Cast.

Casts contents to another object type.

Template Parameters

<i>OtherCrtp</i>	Other derived CRTP class.
------------------	---------------------------

Returns

Casted copy.

6.5.3.7 template<class Crtp , class... Types> Crtp magrathea::AbstractAboutObject< Crtp, Types >::copy() const [inline]

Copy.

Generates a copy of the object.

Returns

Copy.

**6.5.3.8 template<class Crtp , class... Types> template<class... Dummy, class Type , class , class > Type &
magrathea::AbstractAboutObject< Crtp, Types >::data(Dummy...) [inline]**

Unified data access.

Unified data inner component access.

Unified data component access.

Provides a direct access to the data.

Template Parameters

<i>Dummy</i>	(Dummy types.)
<i>Type</i>	(Data std::tuple<Types...> type.)

Returns

Reference to the data.

Provides a direct access to the specified component of the data.

Template Parameters

<i>Index</i>	Index of the component.
<i>Dummy</i>	(Dummy types.)
<i>Type</i>	(Component type.)

Returns

Reference to the component of the data.

Provides a direct access to the specified inner component of the specified component of the data.

Template Parameters

<i>Index</i>	Index of the component.
<i>Subscript</i>	Subscript of the inner component.
<i>Dummy</i>	(Dummy types.)
<i>Type</i>	(Inner component type.)

Returns

Reference to the inner component of the data.

6.5.3.9 template<class Crtp , class... Types> template<class... Dummy, class Type , class , class > const Type & magrathea::AbstractAboutObject< Crtp, Types >::data(Dummy...) const [inline]

Unified data getter.

Gets the data.

Template Parameters

<i>Dummy</i>	(Dummy types.)
<i>Type</i>	(Data std::tuple<Types...> type.)

Returns

Immutable reference to the data.

6.5.3.10 template<class Crtp , class... Types> template<unsigned int Index, unsigned int Subscript, class Type , class , class , class > Crtp & magrathea::AbstractAboutObject< Crtp, Types >::data(const Type & value) [inline]

Unified data setter.

Unified data inner component setter.

Unified data component setter.

Sets the data.

Parameters

in	<i>value</i>	Data value.
----	--------------	-------------

Returns

Self reference.

Sets the specified component of the data.

Template Parameters

<i>Index</i>	Index of the component.
--------------	-------------------------

Parameters

<i>in</i>	<i>value</i>	Component value.
-----------	--------------	------------------

Returns

Self reference.

Sets the specified inner component of the specified component of the data.

Template Parameters

<i>Index</i>	Index of the component.
<i>Subscript</i>	Subscript of the inner component.

Parameters

<i>in</i>	<i>value</i>	Inner component value.
-----------	--------------	------------------------

Returns

Self reference.

6.5.3.11 template<class Crtp , class... Types> Crtp & magrathea::AbstractAboutObject< Crtp, Types >::nullify ()
[inline]

Nullify.

Resets all data members to their default values.

Returns

Self reference.

6.5.3.12 template<class Crtp , class... Types> template<class OtherCrtp , class... OtherTypes> bool
magrathea::AbstractAboutObject< Crtp, Types >::operator!= (const AbstractAboutObject< OtherCrtp,
OtherTypes...> & rhs) const [inline]

Not equal to.

Compares for difference and returns true if the contents is different.

Template Parameters

<i>OtherCrtp</i>	(Other derived CRTP class.)
<i>OtherTypes</i>	(Other variadic list of components types.)

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

True if not equal, false if equal.

6.5.3.13 template<class Crtp, class... Types> Crtp & magrathea::AbstractAboutObject< Crtp, Types >::operator= (const AbstractAboutObject< Crtp, Types...> & *rhs*) [inline]

Copy assignment operator.

Assigns contents from the same type of object.

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Self reference.

6.5.3.14 template<class Crtp , class... Types> template<class OtherCrtp , class... OtherTypes> Crtp & magrathea::AbstractAboutObject< Crtp, Types >::operator= (const AbstractAboutObject< OtherCrtp, OtherTypes...> & *rhs*) [inline]

Conversion assignment operator.

Assigns contents from another type of object.

Template Parameters

<i>OtherCrtp</i>	(Other derived CRTP class.)
<i>OtherTypes</i>	(Other variadic list of components types.)

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Self reference.

6.5.3.15 template<class Crtp , class... Types> template<class... OtherTypes> Crtp & magrathea::AbstractAboutObject< Crtp, Types >::operator= (const std::tuple< OtherTypes...> & *rhs*) [inline]

Data assignment operator.

Assigns contents from data.

Template Parameters

<i>OtherTypes</i>	(Other variadic list of object property types.)
-------------------	---

Parameters

in	rhs	Right-hand side.
----	-----	------------------

Returns

Self reference.

6.5.3.16 template<class Crtp , class... Types> template<class OtherCrtp , class... OtherTypes> bool magrathea::AbstractAboutObject< Crtp, Types >::operator== (const AbstractAboutObject< OtherCrtp, OtherTypes...> & rhs) const [inline]

Equal to.

Compares for equality and returns true if the contents is equal.

Template Parameters

<i>OtherCrtp</i>	(Other derived CRTP class.)
<i>OtherTypes</i>	(Other variadic list of components types.)

Parameters

in	rhs	Right-hand side.
----	-----	------------------

Returns

True if equal, false if not equal.

The documentation for this class was generated from the following file:

- /data/home/mbreton/magrathea_pathfinder/src/magrathea/abstractaboutobject.h

6.6 magrathea::AbstractContents< Crtp, Category, Types > Class Template Reference

Tuple abstraction of numerical simulation contents.

```
#include <abstractcontents.h>
```

Public Member Functions

Lifecycle

- [AbstractContents \(\)](#)
Implicit empty constructor.
- [template<class OtherCrtp , class OtherCategory , class... OtherTypes> AbstractContents \(const AbstractContents< OtherCrtp, OtherCategory, OtherTypes...> &source\)](#)
Explicit conversion constructor.
- [template<class... OtherTypes> AbstractContents \(const std::tuple< OtherTypes...> &source\)](#)
Explicit data constructor.
- [template<class... OtherTypes, class = typename std::enable_if<\(sizeof...\(OtherTypes\) != 0\) && \(std::is_constructible<typename std::tuple_element<0, typename std::conditional<sizeof...\(Types\) != 0, std::tuple<Types...>, std::tuple<std::true_type> >::type>::type, typename std::tuple_element<0, typename std::conditional<sizeof...\(OtherTypes\) != 0, std::tuple<OtherTypes...>, std::tuple<std::true_type> >::type>::type>::value>::type>::type> AbstractContents \(const OtherTypes &...source\)](#)

Explicit components constructor.

Operators

- `Crtp & operator= (const AbstractContents< Crtp, Category, Types...> &rhs)`
Copy assignment operator.
- `template<class OtherCrtp , class OtherCategory , class... OtherTypes>`
`Crtp & operator= (const AbstractContents< OtherCrtp, OtherCategory, OtherTypes...> &rhs)`
Conversion assignment operator.
- `template<class... OtherTypes>`
`Crtp & operator= (const std::tuple< OtherTypes...> &rhs)`
Data assignment operator.
- `template<class OtherCrtp , class OtherCategory , class... OtherTypes>`
`bool operator==(const AbstractContents< OtherCrtp, OtherCategory, OtherTypes...> &rhs) const`
Equal to.
- `template<class OtherCrtp , class OtherCategory , class... OtherTypes>`
`bool operator!=(const AbstractContents< OtherCrtp, OtherCategory, OtherTypes...> &rhs) const`
Not equal to.

Assignment

- `Crtp & assign ()`
Empty assignment.
- `Crtp & assign (const AbstractContents< Crtp, Category, Types...> &source)`
Copy assignment.
- `template<class OtherCrtp , class OtherCategory , class... OtherTypes>`
`Crtp & assign (const AbstractContents< OtherCrtp, OtherCategory, OtherTypes...> &source)`
Conversion assignment.
- `template<class... OtherTypes>`
`Crtp & assign (const std::tuple< OtherTypes...> &source)`
Data assignment.
- `template<class... OtherTypes, class = typename std::enable_if<(sizeof...(OtherTypes) != 0) && (std::is_constructible<typename std::tuple_element<0, typename std::conditional<sizeof...(Types) != 0, std::tuple<Types...>, std::tuple<std::true_type> >::type>-::type, typename std::tuple_element<0, typename std::conditional<sizeof...(OtherTypes) != 0, std::tuple<OtherTypes...>, std::tuple<std::true_type> >::type>::type>::value>::type>`
`Crtp & assign (const OtherTypes &...source)`
Components assignment.

Management

- `Crtp & nullify ()`
Nullify.
- `Crtp copy () const`
Copy.
- `template<class OtherCrtp = Crtp, class = typename std::enable_if<std::is_constructible<OtherCrtp, Crtp>::value>::type>`
`OtherCrtp cast () const`
Cast.

Data

- `template<class... Dummy, class Type = typename std::conditional<sizeof...(Dummy) == 0, std::tuple<Types...>, void>::type, class = typename std::enable_if<sizeof...(Dummy) == 0>::type, class = typename std::enable_if<std::is_convertible<Type, typename std::conditional<sizeof...(Dummy) == 0, std::tuple<Types...>, void>::type>::value>::type>`
`Type & data (Dummy...)`
Unified data access.
- `template<class... Dummy, class Type = typename std::conditional<sizeof...(Dummy) == 0, std::tuple<Types...>, void>::type, class = typename std::enable_if<sizeof...(Dummy) == 0>::type, class = typename std::enable_if<std::is_convertible<Type, typename std::conditional<sizeof...(Dummy) == 0, std::tuple<Types...>, void>::type>::value>::type>`
`const Type & data (Dummy...) const`

- Unified data getter.*
- template<class Type , class = typename std::enable_if<std::is_convertible<Type, typename std::conditional<!std::is_void<Type>->::value, std::tuple<Types...>, void>::type>::value>::type>
- Crtp & **data** (const Type &value)
- Unified data setter.*

Protected Member Functions

Protected lifecycle

- **~AbstractContents ()**
- Protected destructor.*

6.6.1 Detailed Description

```
template<class Crtp, class Category, class... Types>class magrathea::AbstractContents< Crtp, Category, Types >
```

Tuple abstraction of numerical simulation contents.

This class is an abstraction of physical contents in a simulation. It can be a lagrangian, an eulerian or a grid quantity or even other non-traditional type. This abstraction provides standardized access to generic types of internal data that can represents an index, a position, a velocity vector, a temperature, a density or everything else... Mixed-types operations are provided for objects of the same category.

Template Parameters

<i>Crtp</i>	Derived CRTP class.
<i>Category</i>	Contents category (Lagrangian, Eulerian, Grid...).
<i>Types</i>	Variadic list of components types.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 template<class Crtp , class Category , class... Types> **magrathea::AbstractContents< Crtp, Category, Types >::~AbstractContents()** [inline], [protected], [default]

Protected destructor.

Avoids direct instantiation of the class, and only allows it through its derived children.

6.6.2.2 template<class Crtp , class Category , class... Types> **magrathea::AbstractContents< Crtp, Category, Types >::AbstractContents()** [inline]

Implicit empty constructor.

Provides an implicit construction of an object initialized to its default value.

6.6.2.3 template<class Crtp , class Category , class... Types> template<class OtherCrtp , class OtherCategory , class... OtherTypes> **magrathea::AbstractContents< Crtp, Category, Types >::AbstractContents(const AbstractContents< OtherCrtp, OtherCategory, OtherTypes...> & source)** [inline], [explicit]

Explicit conversion constructor.

Provides an explicit construction from another type of object.

Template Parameters

<i>OtherCrtp</i>	(Other derived CRTP class.)
<i>OtherCategory</i>	(Other contents category (Lagrangian, Eulerian, Grid...).)
<i>OtherTypes</i>	(Other variadic list of components types.)

Parameters

<i>in</i>	<i>source</i>	Source of the copy.
-----------	---------------	---------------------

6.6.2.4 template<class Crtp , class Category , class... Types> template<class... OtherTypes>
magrathea::AbstractContents< Crtp, Category, Types >::AbstractContents (const std::tuple<
*OtherTypes...***> & source) [inline], [explicit]**

Explicit data constructor.

Provides an explicit construction from data.

Template Parameters

<i>OtherTypes</i>	(Other variadic list of object property types.)
-------------------	---

Parameters

<i>in</i>	<i>source</i>	Source of the copy.
-----------	---------------	---------------------

6.6.2.5 template<class Crtp , class Category , class... Types> template<class... OtherTypes, class >
magrathea::AbstractContents< Crtp, Category, Types >::AbstractContents (const OtherTypes &... source)
[inline], [explicit]

Explicit components constructor.

Provides an explicit construction from components.

Template Parameters

<i>OtherTypes</i>	(Other variadic list of object property types.)
-------------------	---

Parameters

<i>in</i>	<i>source</i>	Source of the copy.
-----------	---------------	---------------------

6.6.3 Member Function Documentation

6.6.3.1 template<class Crtp , class Category , class... Types> Crtp & **magrathea::AbstractContents< Crtp, Category,**
Types >::assign () [inline]

Empty assignment.

Assigns contents from an object initialized to its default value.

Returns

Self reference.

6.6.3.2 template<class Crtp, class Category, class... Types> Crtp & magrathea::AbstractContents< Crtp, Category, Types >::assign (const AbstractContents< Crtp, Category, Types...> & source) [inline]

Copy assignment.

Assigns contents from the same type of object.

Parameters

in	source	Source of the copy.
----	--------	---------------------

Returns

Self reference.

6.6.3.3 template<class Crtp , class Category , class... Types> template<class OtherCrtp , class OtherCategory , class... OtherTypes> Crtp & magrathea::AbstractContents< Crtp, Category, Types >::assign (const AbstractContents< OtherCrtp, OtherCategory, OtherTypes...> & source) [inline]

Conversion assignment.

Assigns contents from another type of object.

Template Parameters

<i>OtherCrtp</i>	(Other derived CRTP class.)
<i>OtherCategory</i>	(Other contents category (Lagrangian, Eulerian, Grid...).)
<i>OtherTypes</i>	(Other variadic list of components types.)

Parameters

in	source	Source of the copy.
----	--------	---------------------

Returns

Self reference.

6.6.3.4 template<class Crtp , class Category , class... Types> template<class... OtherTypes> Crtp & magrathea::AbstractContents< Crtp, Category, Types >::assign (const std::tuple< OtherTypes...> & source) [inline]

Data assignment.

Assigns contents from data.

Template Parameters

<i>OtherTypes</i>	(Other variadic list of object property types.)
-------------------	---

Parameters

in	source	Source of the copy.
----	--------	---------------------

Returns

Self reference.

6.6.3.5 template<class Crtp , class Category , class... Types> template<class... OtherTypes, class > Crtp &
magrathea::AbstractContents< Crtp, Category, Types >::assign (const OtherTypes &... source) [inline]

Components assignment.

Assigns contents from components.

Template Parameters

<i>OtherTypes</i>	(Other variadic list of object property types.)
-------------------	---

Parameters

in	<i>source</i>	Source of the copy.
----	---------------	---------------------

Returns

Self reference.

6.6.3.6 template<class Crtp , class Category , class... Types> template<class OtherCrtp , class > OtherCrtp
magrathea::AbstractContents< Crtp, Category, Types >::cast () const [inline]

Cast.

Casts contents to another object type.

Template Parameters

<i>OtherCrtp</i>	Other derived CRTP class.
------------------	---------------------------

Returns

Casted copy.

6.6.3.7 template<class Crtp , class Category , class... Types> Crtp **magrathea::AbstractContents< Crtp, Category, Types >::copy () const** [inline]

Copy.

Generates a copy of the object.

Returns

Copy.

6.6.3.8 template<class Crtp , class Category , class... Types> template<class... Dummy, class Type , class , class > Type &
magrathea::AbstractContents< Crtp, Category, Types >::data (Dummy...) [inline]

Unified data access.

Unified data inner component access.

Unified data component access.

Provides a direct access to the data.

Template Parameters

<i>Dummy</i>	(Dummy types.)
<i>Type</i>	(Data std::tuple<Types...> type.)

Returns

Reference to the data.

Provides a direct access to the specified component of the data.

Template Parameters

<i>Index</i>	Index of the component.
<i>Dummy</i>	(Dummy types.)
<i>Type</i>	(Component type.)

Returns

Reference to the component of the data.

Provides a direct access to the specified inner component of the specified component of the data.

Template Parameters

<i>Index</i>	Index of the component.
<i>Subscript</i>	Subscript of the inner component.
<i>Dummy</i>	(Dummy types.)
<i>Type</i>	(Inner component type.)

Returns

Reference to the inner component of the data.

6.6.3.9 template<class Crtp , class Category , class... Types> template<class... Dummy, class Type , class , class > const Type & magrathea::AbstractContents< Crtp, Category, Types >::data (Dummy...) const [inline]

Unified data getter.

Gets the data.

Template Parameters

<i>Dummy</i>	(Dummy types.)
<i>Type</i>	(Data std::tuple<Types...> type.)

Returns

Immutable reference to the data.

6.6.3.10 template<class Crtp , class Category , class... Types> template<unsigned int Index, unsigned int Subscript, class Type , class , class , class > Crtp & magrathea::AbstractContents< Crtp, Category, Types >::data (const Type & value) [inline]

Unified data setter.

Unified data inner component setter.

Unified data component setter.

Sets the data.

Parameters

in	<i>value</i>	Data value.
----	--------------	-------------

Returns

Self reference.

Sets the specified component of the data.

Template Parameters

<i>Index</i>	Index of the component.
--------------	-------------------------

Parameters

in	<i>value</i>	Component value.
----	--------------	------------------

Returns

Self reference.

Sets the specified inner component of the specified component of the data.

Template Parameters

<i>Index</i>	Index of the component.
<i>Subscript</i>	Subscript of the inner component.

Parameters

in	<i>value</i>	Inner component value.
----	--------------	------------------------

Returns

Self reference.

6.6.3.11 template<class Crtp , class Category , class... Types> Crtp & magrathea::AbstractContents< Crtp, Category, Types >::nullify() [inline]

Nullify.

Resets all data members to their default values.

Returns

Self reference.

6.6.3.12 template<class Crtp , class Category , class... Types> template<class OtherCrtp , class OtherCategory , class... OtherTypes> bool magrathea::AbstractContents< Crtp, Category, Types >::operator!= (const AbstractContents< OtherCrtp, OtherCategory, OtherTypes...> & rhs) const [inline]

Not equal to.

Compares for difference and returns true if the contents is different.

Template Parameters

<i>OtherCrt</i>	(Other derived CRTP class.)
<i>OtherCategory</i>	(Other contents category (Lagrangian, Eulerian, Grid...).)
<i>OtherTypes</i>	(Other variadic list of components types.)

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

True if not equal, false if equal.

6.6.3.13 template<class Crtp, class Category, class... Types> Crtp & magrathea::AbstractContents< Crtp, Category, Types >::operator= (const AbstractContents< Crtp, Category, Types...> & rhs) [inline]

Copy assignment operator.

Assigns contents from the same type of object.

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Self reference.

6.6.3.14 template<class Crtp , class Category , class... Types> template<class OtherCrt , class OtherCategory , class... OtherTypes> Crtp & magrathea::AbstractContents< Crtp, Category, Types >::operator= (const AbstractContents< OtherCrt, OtherCategory, OtherTypes...> & rhs) [inline]

Conversion assignment operator.

Assigns contents from another type of object.

Template Parameters

<i>OtherCrt</i>	(Other derived CRTP class.)
<i>OtherCategory</i>	(Other contents category (Lagrangian, Eulerian, Grid...).)
<i>OtherTypes</i>	(Other variadic list of components types.)

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Self reference.

6.6.3.15 template<class Crtp , class Category , class... Types> template<class... OtherTypes> Crtp & magrathea::AbstractContents< Crtp, Category, Types >::operator= (const std::tuple< OtherTypes...> & rhs) [inline]

Data assignment operator.

Assigns contents from data.

Template Parameters

<i>OtherTypes</i>	(Other variadic list of object property types.)
-------------------	---

Parameters

<i>in</i>	<i>rhs</i>	Right-hand side.
-----------	------------	------------------

Returns

Self reference.

6.6.3.16 template<class Crtp , class Category , class... Types> template<class OtherCrtp , class OtherCategory , class... OtherTypes> bool magrathea::AbstractContents< Crtp, Category, Types >::operator== (const AbstractContents< OtherCrtp, OtherCategory, OtherTypes...> & rhs) const [inline]

Equal to.

Compares for equality and returns true if the contents is equal.

Template Parameters

<i>OtherCrtp</i>	(Other derived CRTP class.)
<i>OtherCategory</i>	(Other contents category (Lagrangian, Eulerian, Grid...).)
<i>OtherTypes</i>	(Other variadic list of components types.)

Parameters

<i>in</i>	<i>rhs</i>	Right-hand side.
-----------	------------	------------------

Returns

True if equal, false if not equal.

The documentation for this class was generated from the following file:

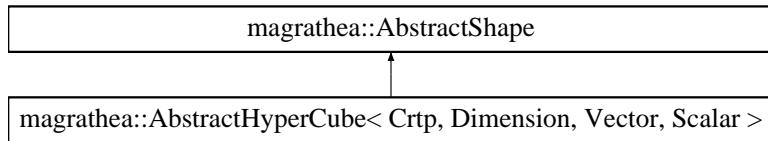
- /data/home/mbreton/magrathea_pathfinder/src/magrathea/abstractcontents.h

6.7 magrathea::AbstractHyperCube< Crtp, Dimension, Vector, Scalar > Class Template Reference

Abstract function provider for n-dimensional cubes.

```
#include <abstracthypercube.h>
```

Inheritance diagram for magrathea::AbstractHyperCube< Crtp, Dimension, Vector, Scalar >:



Public Member Functions

Position

- `constexpr Scalar center (const unsigned int idim) const`
Center coordinate.
- `Vector center () const`
Center vector.
- `constexpr Scalar minimum (const unsigned int idim) const`
Minimum coordinate.
- `Vector minimum () const`
Minimum vector.
- `constexpr Scalar maximum (const unsigned int idim) const`
Maximum coordinate.
- `Vector maximum () const`
Maximum vector.

Measures

- `constexpr Scalar length () const`
Length.
- `constexpr Scalar volume () const`
Volume.
- `template<unsigned int Subdimension = Dimension-(Dimension != 0)>`
`constexpr Scalar surface () const`
Outer surface.
- `template<unsigned int Subdimension = Dimension-(Dimension != 0)>`
`constexpr Scalar area () const`
Element area.
- `template<unsigned int Subdimension = Dimension>`
`constexpr Scalar diagonal () const`
Diagonal.

Distribution

- `template<unsigned int Subdimension = Dimension, class = typename std::enable_if<(Subdimension != 0) || (Subdimension == 0)>-::type>`
`Vector random () const`
Basic random location in the hypercube.
- `template<unsigned int Subdimension = Dimension, class Engine , class Distribution , class = typename std::enable_if<((Subdimension != 0) || (Subdimension == 0)) && (std::decay<Engine>::type::min() != std::decay<Engine>::type::max()) && (!std::is_void<typename std::decay<Distribution>::type::result_type>::value)>::type>`
`Vector random (Engine &engine, Distribution &&distribution) const`
Generic random location in the hypercube.

Collision

- template<class OtherVector , class = typename std::enable_if<std::is_convertible<typename std::remove_cv<typename std::remove_reference<decltype(std::declval<OtherVector>()[0])>::type>::type, Scalar>::value>::type>
bool **inside** (const OtherVector &**point**) const
Point inside.
- template<class OtherVector , class = typename std::enable_if<std::is_convertible<typename std::remove_cv<typename std::remove_reference<decltype(std::declval<OtherVector>()[0])>::type>::type, Scalar>::value>::type>
bool **outside** (const OtherVector &**point**) const
Point outside.

Static Public Member Functions

Constants

- static constexpr unsigned int **dimension** ()
Number of space dimension.
- template<unsigned int Subdimension = Dimension>
static constexpr unsigned int **elements** ()
Number of elements.
- template<unsigned int FirstSubdimension = 0, unsigned int LastSubdimension = Dimension-(Dimension != 0)>
static constexpr unsigned int **subelements** ()
Sum of elements.

Test

- static int **example** ()
Example function.

Protected Member Functions

Protected lifecycle

- ~AbstractHyperCube** ()
Protected destructor.

6.7.1 Detailed Description

template<class Crtp, unsigned int Dimension, class Vector, typename Scalar>class **magrathea::AbstractHyperCube< Crtp, Dimension, Vector, Scalar >**

Abstract function provider for n-dimensional cubes.

Provides a common base for n-dimensional cubes thanks to CRTP. To use it, one has to derive from this class and pass the derived class itself as the CRTP parameter. The derived classes should provide two immutable functions :

- position()**
- extent()**

in order to get the position of the center along one coordinate and the edge length of the hypercube.

Template Parameters

<i>Crtp</i>	Derived CRTP class.
<i>Dimension</i>	Number of space dimension.
<i>Vector</i>	Position vector type.
<i>Scalar</i>	Scalar data type.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 template<class Crtp , unsigned int Dimension, class Vector , typename Scalar > **magrathea::AbstractHyperCube< Crtp, Dimension, Vector, Scalar >::~AbstractHyperCube()** [inline], [protected], [default]

Protected destructor.

Avoids direct instantiation of the class, and only allows it through its derived children.

6.7.3 Member Function Documentation

6.7.3.1 template<class Crtp , unsigned int Dimension, class Vector , typename Scalar > template<unsigned int Subdimension> constexpr Scalar **magrathea::AbstractHyperCube< Crtp, Dimension, Vector, Scalar >::area() const**

Element area.

Computes the surface of a single element of the hypercube.

Template Parameters

<i>Subdimension</i>	Dimension of elements.
---------------------	------------------------

Returns

l^D .

6.7.3.2 template<class Crtp , unsigned int Dimension, class Vector , typename Scalar > constexpr Scalar **magrathea::AbstractHyperCube< Crtp, Dimension, Vector, Scalar >::center(const unsigned int *idim*) const**

Center coordinate.

Computes the specified coordinate of the center of the hypercube.

Parameters

in	<i>idim</i>	Index of the dimension.
----	-------------	-------------------------

Returns

The coordinate x_i .

6.7.3.3 template<class Crtp , unsigned int Dimension, class Vector , typename Scalar > Vector **magrathea::AbstractHyperCube< Crtp, Dimension, Vector, Scalar >::center() const** [inline]

Center vector.

Computes the position vector of the center of the hypercube.

Returns

The position vector \vec{x} .

6.7.3.4 template<class Crtp , unsigned int Dimension, class Vector , typename Scalar > template<unsigned int Subdimension> constexpr Scalar magrathea::AbstractHyperCube< Crtp, Dimension, Vector, Scalar >::diagonal() const

Returns

Computes the length of the diagonal in the specified subdimension.

Template Parameters

<i>Subdimension</i>	Dimension of elements.
---------------------	------------------------

Returns

$$\sqrt{D} \times l.$$

6.7.3.5 template<class Crtp , unsigned int Dimension, class Vector , typename Scalar > constexpr unsigned int magrathea::AbstractHyperCube< Crtp, Dimension, Vector, Scalar >::dimension() [static]

Returns

Computes the number of space dimension of the hypercube.

Returns

Dimension.

6.7.3.6 template<class Crtp , unsigned int Dimension, class Vector , typename Scalar > template<unsigned int Subdimension> constexpr unsigned int magrathea::AbstractHyperCube< Crtp, Dimension, Vector, Scalar >::elements() [static]

Returns

Computes the number of elements of the specified subdimension in the hypercube. For example, for a specified subdimension 0, it computes the number of points.

Template Parameters

<i>Subdimension</i>	Dimension of elements.
---------------------	------------------------

Returns

$$2^{D-d} D d.$$

6.7.3.7 template<class Crtp , unsigned int Dimension, class Vector , typename Scalar > int magrathea::AbstractHyperCube< Crtp, Dimension, Vector, Scalar >::example() [static]

Returns

Tests and demonstrates the use of [AbstractHyperCube](#).

Returns

0 if no error.

6.7.3.8 template<class Crtp , unsigned int Dimension, class Vector , typename Scalar > template<class OtherVector , class >
bool magrathea::AbstractHyperCube< Crtp, Dimension, Vector, Scalar >::inside (const OtherVector & point)
const [inline]

Point inside.

Checks whether a point is inside the hypercube.

Template Parameters

<i>OtherVector</i>	Other position vector type.
--------------------	-----------------------------

Parameters

<i>in</i>	<i>point</i>	Position of the point.
-----------	--------------	------------------------

Returns

True if the point is inside the hypercube, false otherwise.

6.7.3.9 template<class Crtp , unsigned int Dimension, class Vector , typename Scalar > constexpr Scalar
magrathea::AbstractHyperCube< Crtp, Dimension, Vector, Scalar >::length () const

Length.

Computes the edge length of the hypercube.

Returns

l.

6.7.3.10 template<class Crtp , unsigned int Dimension, class Vector , typename Scalar > constexpr Scalar
magrathea::AbstractHyperCube< Crtp, Dimension, Vector, Scalar >::maximum (const unsigned int idim)
const

Maximum coordinate.

Computes the specified coordinate of the maximum boundary of the hypercube.

Parameters

<i>in</i>	<i>idim</i>	Index of the dimension.
-----------	-------------	-------------------------

Returns

The coordinate $x_i + \frac{l}{2}$.

6.7.3.11 template<class Crtp , unsigned int Dimension, class Vector , typename Scalar > Vector
magrathea::AbstractHyperCube< Crtp, Dimension, Vector, Scalar >::maximum () const [inline]

Maximum vector.

Computes the position vector of the maximum boundary of the hypercube.

Returns

The position vector $\vec{x} + \frac{\vec{l}}{2}$.

6.7.3.12 template<class Crtp , unsigned int Dimension, class Vector , typename Scalar > constexpr Scalar
magrathea::AbstractHyperCube< Crtp, Dimension, Vector, Scalar >::minimum (const unsigned int *idim*) const

Minimum coordinate.

Computes the specified coordinate of the minimum boundary of the hypercube.

Parameters

in	<i>idim</i>	Index of the dimension.
----	-------------	-------------------------

Returns

The coordinate $x_i - \frac{l}{2}$.

6.7.3.13 template<class Crtp , unsigned int Dimension, class Vector , typename Scalar > Vector
magrathea::AbstractHyperCube< Crtp, Dimension, Vector, Scalar >::minimum () const [inline]

Minimum vector.

Computes the position vector of the minimum boundary of the hypercube.

Returns

The position vector $\vec{x} - \frac{\vec{l}}{2}$.

6.7.3.14 template<class Crtp , unsigned int Dimension, class Vector , typename Scalar > template<class OtherVector , class > bool **magrathea::AbstractHyperCube< Crtp, Dimension, Vector, Scalar >::outside (const OtherVector & *point*) const [inline]**

Point outside.

Checks whether a point is outside the hypercube.

Template Parameters

OtherVector	Other position vector type.
-------------	-----------------------------

Parameters

in	<i>point</i>	Position of the point.
----	--------------	------------------------

Returns

True if the point is outside the hypercube, false otherwise.

6.7.3.15 template<class Crtp , unsigned int Dimension, class Vector , typename Scalar > template<unsigned int Subdimension, class > Vector **magrathea::AbstractHyperCube< Crtp, Dimension, Vector, Scalar >::random () const**

Basic random location in the hypercube.

Generates a random location, located on the subdimensional elements of the hypercube. For example for a subdimension of 2 of a 3-dimensional hypercube, the function will generate a random point located on the surface of the cube.

Template Parameters

<i>Subdimension</i>	Dimension space.
---------------------	------------------

Returns

Random position vector.

Warning

As the internal engine is a static one, do not use this function in parallel.

6.7.3.16 `template<class Crtp , unsigned int Dimension, class Vector , typename Scalar > template<unsigned int Subdimension, class Engine , class Distribution , class > Vector magrathea::AbstractHyperCube< Crtp, Dimension, Vector, Scalar >::random (Engine && engine, Distribution && distribution) const`

Generic random location in the hypercube.

Generates a random location, located on the subdimensional elements of the hypercube. For example for a subdimension of 2 of a 3-dimensional hypercube, the function will generates a random point located on the surface of the cube. As this function uses the passed random engine and distribution it is completely thread safe.

Template Parameters

<i>Subdimension</i>	Dimension space.
<i>Engine</i>	(Random engine type.)
<i>Distribution</i>	(Random distribution type.)

Parameters

<i>in, out</i>	<i>engine</i>	Random engine.
<i>in, out</i>	<i>distribution</i>	Random distribution.

Returns

Random position vector.

6.7.3.17 `template<class Crtp , unsigned int Dimension, class Vector , typename Scalar > template<unsigned int First, unsigned int Last> constexpr unsigned int magrathea::AbstractHyperCube< Crtp, Dimension, Vector, Scalar >::subelements () [static]`

Sum of elements.

Computes the sum of the hypercube elements between the two given dimensions.

Template Parameters

<i>First</i>	First dimension of elements.
<i>Last</i>	Last dimension of elements.

Returns

$\sum_d 2^{D-d} Dd.$

6.7.3.18 template<class Crtp , unsigned int Dimension, class Vector , typename Scalar > template<unsigned int Subdimension> constexpr Scalar **magrathea::AbstractHyperCube< Crtp, Dimension, Vector, Scalar >**::surface () const

Outer surface.

Computes the total outer surface of the hypercube.

Template Parameters

<i>Subdimension</i>	Dimension of elements.
---------------------	------------------------

Returns

$2^{D-d} D d \times l^D$.

6.7.3.19 template<class Crtp , unsigned int Dimension, class Vector , typename Scalar > constexpr Scalar **magrathea::AbstractHyperCube< Crtp, Dimension, Vector, Scalar >**::volume () const

Volume.

Computes the volume of the hypercube.

Returns

l^D .

The documentation for this class was generated from the following file:

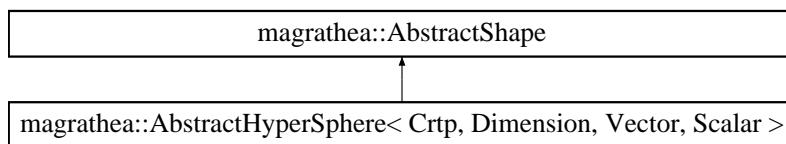
- /data/home/mbreton/magrathea_pathfinder/src/magrathea/abstracthypercube.h

6.8 **magrathea::AbstractHyperSphere< Crtp, Dimension, Vector, Scalar >** Class Template Reference

Abstract function provider for n-dimensional spheres.

```
#include <abstracthypersphere.h>
```

Inheritance diagram for **magrathea::AbstractHyperSphere< Crtp, Dimension, Vector, Scalar >**:



Public Member Functions

Position

- constexpr Scalar **center** (const unsigned int idim) const
Center coordinate.
- Vector **center** () const
Center vector.
- constexpr Scalar **minimum** (const unsigned int idim) const

- Minimum coordinate.
- Vector **minimum** () const
 - Minimum vector.*
- constexpr Scalar **maximum** (const unsigned int idim) const
 - Maximum coordinate.*
- Vector **maximum** () const
 - Maximum vector.*

Measures

- constexpr Scalar **radius** () const
 - Radius.*
- constexpr Scalar **diameter** () const
 - Diameter.*
- constexpr Scalar **volume** () const
 - Volume.*
- constexpr Scalar **surface** () const
 - Outer surface.*

Distribution

- template<unsigned int Subdimension = Dimension, class = typename std::enable_if<(Subdimension+1 == Dimension) || (-Subdimension == Dimension)>::type>
 - Vector **random** () const
 - Basic random location in the hypersphere.*
- template<unsigned int Subdimension = Dimension, class Engine , class Distribution , class = typename std::enable_if<((-Subdimension+1 == Dimension) || (Subdimension == Dimension)) && (std::decay<Engine>::type::min() != std::decay<Engine>::type::max()) && (!std::is_void<typename std::decay<Distribution>::type::result_type>::value)>::type>
 - Vector **random** (Engine &&engine, Distribution &&distribution) const
 - Generic random location in the hypersphere.*
- template<unsigned int Subdimension = Dimension, typename Iterator , class = typename std::enable_if<(Subdimension+1 == Dimension) && (Subdimension >= 1)>::type std::pair< Scalar, Scalar > **uniform** (const Iterator &first, const Iterator &last) const
 - Uniform distribution.*

Collision

- template<class OtherVector , class = typename std::enable_if<std::is_convertible<typename std::remove_cv<typename std::remove_reference<decltype(std::declval<OtherVector>()[0])>::type>::type, Scalar>::value>::type>
 - bool **inside** (const OtherVector &**point**) const
 - Point inside.*
- template<class OtherVector , class = typename std::enable_if<std::is_convertible<typename std::remove_cv<typename std::remove_reference<decltype(std::declval<OtherVector>()[0])>::type>::type, Scalar>::value>::type>
 - bool **outside** (const OtherVector &**point**) const
 - Point outside.*

Static Public Member Functions

Constants

- static constexpr unsigned int **dimension** ()
 - Number of space dimension.*

Helpers

- template<unsigned int OtherDimension = Dimension, typename OtherScalar = Scalar>
 - static constexpr OtherScalar **sn** ()

Surface of a n-dimensional unit sphere.

Test

- static int [example \(\)](#)
Example function.

Protected Member Functions

Protected lifecycle

- [~AbstractHyperSphere \(\)](#)
Protected destructor.

6.8.1 Detailed Description

```
template<class Crtp, unsigned int Dimension, class Vector, typename Scalar>class magrathea::AbstractHyperSphere< Crtp, Dimension, Vector, Scalar >
```

Abstract function provider for n-dimensional spheres.

Provides a common base for n-dimensional spheres thanks to CRTP. To use it, one has to derive from this class and pass the derived class itself as the CRTP parameter. The derived classes should provide two immutable functions :

- [position\(\)](#)
- [extent\(\)](#)

in order to get the position of the center along one coordinate and the radius length of the hypersphere.

Template Parameters

<i>Crtp</i>	Derived CRTP class.
<i>Dimension</i>	Number of space dimension.
<i>Vector</i>	Position vector type.
<i>Scalar</i>	Scalar data type.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 template<class Crtp , unsigned int Dimension, class Vector , typename Scalar > [magrathea::AbstractHyperSphere< Crtp, Dimension, Vector, Scalar >::~AbstractHyperSphere \(\)](#) [inline], [protected], [default]

Protected destructor.

Avoids direct instantiation of the class, and only allows it through its derived children.

6.8.3 Member Function Documentation

6.8.3.1 template<class Crtp , unsigned int Dimension, class Vector , typename Scalar > [constexpr Scalar magrathea::AbstractHyperSphere< Crtp, Dimension, Vector, Scalar >::center \(const unsigned int idim \) const](#)

Center coordinate.

Computes the specified coordinate of the center of the hypersphere.

Parameters

in	<i>idim</i>	Index of the dimension.
----	-------------	-------------------------

Returns

The coordinate x_i .

**6.8.3.2 template<class Crtp , unsigned int Dimension, class Vector , typename Scalar > Vector
magrathea::AbstractHyperSphere< Crtp, Dimension, Vector, Scalar >::center() const [inline]**

Center vector.

Computes the position vector of the center of the hypersphere.

Returns

The position vector \vec{x} .

**6.8.3.3 template<class Crtp , unsigned int Dimension, class Vector , typename Scalar > constexpr Scalar
magrathea::AbstractHyperSphere< Crtp, Dimension, Vector, Scalar >::diameter() const**

Diameter.

Computes the diameter of the hypersphere.

Returns

$2r$.

**6.8.3.4 template<class Crtp , unsigned int Dimension, class Vector , typename Scalar > constexpr unsigned int
magrathea::AbstractHyperSphere< Crtp, Dimension, Vector, Scalar >::dimension() [static]**

Number of space dimension.

Computes the number of space dimension of the hypercube.

Returns

Dimension.

**6.8.3.5 template<class Crtp , unsigned int Dimension, class Vector , typename Scalar > int
magrathea::AbstractHyperSphere< Crtp, Dimension, Vector, Scalar >::example() [static]**

Example function.

Tests and demonstrates the use of [AbstractHyperSphere](#).

Returns

0 if no error.

6.8.3.6 template<class Crtp , unsigned int Dimension, class Vector , typename Scalar > template<class OtherVector , class > bool **magrathea::AbstractHyperSphere< Crtp, Dimension, Vector, Scalar >::inside** (const OtherVector & *point*)
const [inline]

Point inside.

Checks whether a point is inside the hypersphere.

Template Parameters

<i>OtherVector</i>	Other position vector type.
--------------------	-----------------------------

Parameters

in	<i>point</i>	Position of the point.
----	--------------	------------------------

Returns

True if the point is inside the hypersphere, false otherwise.

6.8.3.7 template<class Crtp , unsigned int Dimension, class Vector , typename Scalar > constexpr Scalar **magrathea::AbstractHyperSphere< Crtp, Dimension, Vector, Scalar >::maximum** (const unsigned int *idim*)
const

Maximum coordinate.

Computes the specified coordinate of the maximum boundary of the hypersphere.

Parameters

in	<i>idim</i>	Index of the dimension.
----	-------------	-------------------------

Returns

The coordinate $x_i + r$.

6.8.3.8 template<class Crtp , unsigned int Dimension, class Vector , typename Scalar > Vector **magrathea::AbstractHyperSphere< Crtp, Dimension, Vector, Scalar >::maximum** () const [inline]

Maximum vector.

Computes the position vector of the maximum boundary of the hypersphere.

Returns

The position vector $\vec{x} + \vec{r}$.

6.8.3.9 template<class Crtp , unsigned int Dimension, class Vector , typename Scalar > constexpr Scalar **magrathea::AbstractHyperSphere< Crtp, Dimension, Vector, Scalar >::minimum** (const unsigned int *idim*)
const

Minimum coordinate.

Computes the specified coordinate of the minimum boundary of the hypersphere.

Parameters

in	<i>idim</i>	Index of the dimension.
----	-------------	-------------------------

Returns

The coordinate $x_i - r$.

**6.8.3.10 template<class Crtp , unsigned int Dimension, class Vector , typename Scalar > Vector
magrathea::AbstractHyperSphere< Crtp, Dimension, Vector, Scalar >::minimum () const [inline]**

Minimum vector.

Computes the position vector of the minimum boundary of the hypersphere.

Returns

The position vector $\vec{x} - \vec{r}$.

**6.8.3.11 template<class Crtp , unsigned int Dimension, class Vector , typename Scalar > template<class OtherVector , class
> bool magrathea::AbstractHyperSphere< Crtp, Dimension, Vector, Scalar >::outside (const OtherVector &
point) const [inline]**

Point outside.

Checks whether a point is outside the hypersphere.

Template Parameters

<i>OtherVector</i>	Other position vector type.
--------------------	-----------------------------

Parameters

in	<i>point</i>	Position of the point.
----	--------------	------------------------

Returns

True if the point is outside the hypersphere, false otherwise.

**6.8.3.12 template<class Crtp , unsigned int Dimension, class Vector , typename Scalar > constexpr Scalar
magrathea::AbstractHyperSphere< Crtp, Dimension, Vector, Scalar >::radius () const**

Radius.

Computes the radius of the hypersphere.

Returns

r.

**6.8.3.13 template<class Crtp , unsigned int Dimension, class Vector , typename Scalar > template<unsigned int
Subdimension, class > Vector magrathea::AbstractHyperSphere< Crtp, Dimension, Vector, Scalar >::random () const**

Basic random location in the hypersphere.

Generates a random location, located in the volume or on the surface of the hypersphere. For example for a subdimension of 2 of a 3-dimensional hypersphere, the function will generates a random point located on the surface of the sphere.

Template Parameters

<i>Subdimension</i>	Dimension space.
---------------------	------------------

Returns

Random position vector.

Warning

As the internal engine is a static one, do not use this function in parallel.

6.8.3.14 template<class Crtp , unsigned int Dimension, class Vector , typename Scalar > template<unsigned int Subdimension, class Engine , class Distribution , class > Vector magrathea::AbstractHyperSphere< Crtp, Dimension, Vector, Scalar >::random (Engine && *engine*, Distribution && *distribution*) const

Generic random location in the hypersphere.

Generates a random location, located in the volume or on the surface of the hypersphere. For example for a subdimension of 2 of a 3-dimensional hypersphere, the function will generates a random point located on the surface of the sphere. As this function uses the passed random engine and distribution it is completely thread safe.

Template Parameters

<i>Subdimension</i>	Dimension space.
<i>Engine</i>	(Random engine type.)
<i>Distribution</i>	(Random distribution type.)

Parameters

<i>in,out</i>	<i>engine</i>	Random engine.
<i>in,out</i>	<i>distribution</i>	Random distribution.

Returns

Random position vector.

6.8.3.15 template<class Crtp , unsigned int Dimension, class Vector , typename Scalar > template<unsigned int OtherDimension, typename OtherScalar > constexpr OtherScalar magrathea::AbstractHyperSphere< Crtp, Dimension, Vector, Scalar >::sn () [static]

Surface of a n-dimensional unit sphere.

Computes the surface of a n-dimensional sphere with a unit radius.

Template Parameters

<i>OtherDimension</i>	Other number of space dimension.
<i>OtherScalar</i>	(Other scalar data type.)

Returns
 s_n .

6.8.3.16 template<class Crtp , unsigned int Dimension, class Vector , typename Scalar > constexpr Scalar magrathea::AbstractHyperSphere< Crtp, Dimension, Vector, Scalar >::surface () const

Outer surface.

Computes the total outer surface of the hypercube.

Returns
 $s_n r^{n-1}$.

6.8.3.17 template<class Crtp , unsigned int Dimension, class Vector , typename Scalar > template<unsigned int Subdimension, typename Iterator , class > std::pair< Scalar, Scalar > magrathea::AbstractHyperSphere< Crtp, Dimension, Vector, Scalar >::uniform (const Iterator & *first*, const Iterator & *last*) const

Uniform distribution.

Generates a random distribution of points located on the surface of the hypersphere. Currently the function only works in two or three dimension corresponding to a value of one or two for the subdimension. In three dimensions, it uses a spiral to approximately distribute the points. The function eventually returns a pair of distance corresponding to the minimum and maximum distances between two generated points.

Template Parameters

<i>Subdimension</i>	Dimension space.
<i>Iterator</i>	(Pointer or iterator to vector types.)

Parameters

in	<i>first</i>	Beginning of the interval.
in	<i>end</i>	End of the interval.

Returns

Minimum and maximum distance between two points.

6.8.3.18 template<class Crtp , unsigned int Dimension, class Vector , typename Scalar > constexpr Scalar magrathea::AbstractHyperSphere< Crtp, Dimension, Vector, Scalar >::volume () const

Volume.

Computes the volume of the hypersphere.

Returns
 $\frac{s_n r^n}{n}$.

The documentation for this class was generated from the following file:

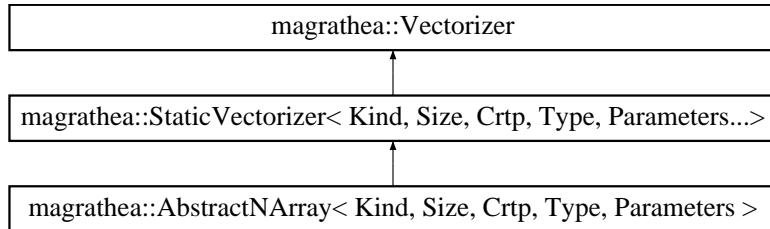
- /data/home/mbreton/magrathea_pathfinder/src/magrathea/[abstracthypersphere.h](#)

6.9 magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters > Class Template Reference

Abstract base class of n-dimensional mathematical arrays.

```
#include <abstractnarray.h>
```

Inheritance diagram for magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >:



Public Member Functions

Lifecycle

- **AbstractNArray ()**
Implicit empty constructor.
- template<typename FundamentalType = Type, class = typename std::enable_if<std::is_fundamental<FundamentalType>::value>::type>
 AbstractNArray (const **AbstractNArray**< Kind, Size, Crtp, FundamentalType, Parameters...> &source)
 Implicit conversion constructor.
- template<typename OtherType = Type, class... Misc, class = typename std::enable_if<std::is_convertible<OtherType, Type>::value>::type>
 AbstractNArray (const std::initializer_list< OtherType > &source, const Misc &...misc)
 Implicit initializer list constructor.
- template<class... Misc, class = typename std::enable_if<sizeof...(Misc) != 0>::type>
 AbstractNArray (const Misc &...misc)
 Explicit generic constructor.

Vectorization

- Type & **operator[]** (const unsigned int i)
Direct access to the element.
- const Type & **operator[]** (const unsigned int i) const
Immutable direct access to the element.

Access

- Type * **data** ()
Direct access to the underlying storage.
- const Type * **data** () const
Immutable direct access to the underlying storage.

Iterators

- Type * **begin** ()
Iterator to the beginning.
- const Type * **begin** () const
Immutable iterator to the beginning.
- const Type * **cbegin** () const

- **Type * end ()**
Iterator to the end.
- **const Type * end () const**
Immutable iterator to the end.
- **const Type * cend () const**
Forced immutable iterator to the end.
- **std::reverse_iterator< Type * > rbegin ()**
Reverse iterator to the beginning.
- **std::reverse_iterator< const Type * > rbegin () const**
Immutable reverse iterator to the beginning.
- **std::reverse_iterator< const Type * > rend ()**
Reverse iterator to the end.
- **std::reverse_iterator< const Type * > rend () const**
Immutable reverse iterator to the end.
- **template<typename IteratorType >**
unsigned int index (const IteratorType &it, typename std::iterator_traits< IteratorType >::iterator_category * = nullptr) const
Index of an iterator in the array.

Comparison

- **bool null (const Type &tolerance) const**
Check whether all elements are approximately null.
- **template<class GenericType >**
bool eq (const GenericType &rhs, const Type &tolerance) const
Compare for approximate equality.
- **template<class GenericType >**
bool ne (const GenericType &rhs, const Type &tolerance) const
Compare for noticeable difference.

Statistics

- **template<class Mask = std::true_type, class = typename std::enable_if<(std::is_base_of<Vectorizer, Mask>::value) || (std::is_same<std::true_type, Mask>::value)>::type>**
const Type & amin (const Mask &bitmask=Mask()) const
Absolute minimum element.
- **template<class Mask = std::true_type, class = typename std::enable_if<(std::is_base_of<Vectorizer, Mask>::value) || (std::is_same<std::true_type, Mask>::value)>::type>**
const Type & amax (const Mask &bitmask=Mask()) const
Maximum element.
- **template<typename Return = typename std::conditional<std::is_floating_point<Type>::value, Type, double>::type, class Coefficient = Type, class Mask = std::true_type, class = typename std::enable_if<(std::is_arithmetic<Return>::value) && ((std::is_base_of<Vectorizer, Coefficient>::value) || (std::is_convertible<Coefficient, Type>::value)) && ((std::is_base_of<Vectorizer, Mask>::value) || (std::is_same<std::true_type, Mask>::value))>::type>**
Return mean (const Coefficient &coefficient=Coefficient(1), const Mask &bitmask=Mask()) const
Mean value.
- **template<typename Return = typename std::conditional<std::is_floating_point<Type>::value, Type, double>::type, typename Correction = Return, class Coefficient = Type, class Mask = std::true_type, class = typename std::enable_if<(std::is_arithmetic<Return>::value) && (std::is_arithmetic<Correction>::value) && ((std::is_base_of<Vectorizer, Coefficient>::value) || (std::is_convertible<Coefficient, Type>::value)) && ((std::is_base_of<Vectorizer, Mask>::value) || (std::is_same<std::true_type, Mask>::value))>::type>**

Return `sigma` (const Correction &correction=Correction(0), const Coefficient &coefficient=Coefficient(1), const Mask &bitmask=Mask()) const

Standard deviation.

Application

- template<typename Return = Type, typename First = Type, class... Types, class... Args, class = typename std::enable_if<(std::is_convertible<typename std::decay<First>::type, typename std::decay<Type>::value) && (sizeof...(Types) == sizeof...(Args))>::type>

`Crtp< Type, Parameters...> & modify (Return(*f)(First, Types...), Args &&...args)`

Modification by a function pointer.

- template<typename Return = Type, typename First = Type, class... Types, class... Args, class Mask , class = typename std::enable_if<(std::is_base_of<Vectorizer, Mask>::value) && (std::is_convertible<typename std::decay<First>::type, typename std::decay<Type>::value) && (sizeof...(Types) == sizeof...(Args))>::type>

`Crtp< Type, Parameters...> & modify (const Mask &bitmask, Return(*f)(First, Types...), Args &&...args)`

Masked modification by a function pointer.

- template<typename Return = Type, typename First = Type, class... Types, class... Args, class = typename std::enable_if<(std::is_convertible<typename std::decay<First>::type, typename std::decay<Type>::value) && (sizeof...(Types) == sizeof...(Args))>::type>

`Crtp< Return, Parameters...> apply (Return(*f)(First, Types...), Args &&...args) const`

Application of a function pointer.

- template<typename Return = Type, typename First = Type, class... Types, class... Args, class Mask , class = typename std::enable_if<(std::is_base_of<Vectorizer, Mask>::value) && (std::is_convertible<typename std::decay<First>::type, typename std::decay<Type>::value) && (sizeof...(Types) == sizeof...(Args))>::type>

`Crtp< Return, Parameters...> apply (const Mask &bitmask, Return(*f)(First, Types...), Args &&...args) const`

Masked application of a function pointer.

Count

- template<class Function = std::equal_to<Type>, class Mask = std::true_type, class = typename std::enable_if<(!std::is_base_of<Vectorizer, typename std::decay<Function>::type>::value) && ((std::is_base_of<Vectorizer, Mask>::value) || (std::is_same<std::true_type, Mask>::value))>::type>

`bool unicity (Function &&f=Function(), const Mask &bitmask=Mask()) const`

Unicity of elements.

- template<class Function = std::equal_to<Type>, class Mask = std::true_type, class = typename std::enable_if<(!std::is_base_of<Vectorizer, typename std::decay<Function>::type>::value) && ((std::is_base_of<Vectorizer, Mask>::value) || (std::is_same<std::true_type, Mask>::value))>::type>

`unsigned int distinct (Function &&f=Function(), const Mask &bitmask=Mask()) const`

Number of distincts elements.

Sort

- template<class Function = std::less<Type>, class Indexes = Type, class Mask = std::true_type, class Pair = std::pair<Type, typename std::remove_const<typename std::remove_reference<decltype(Vectorizer::get(std::declval<Indexes>(), 0))>::type>::type>, class = typename std::enable_if<(!std::is_base_of<Vectorizer, typename std::decay<Function>::type>::value) && ((std::is_base_of<Vectorizer, Mask>::value) || (std::is_same<std::true_type, Mask>::value))>::type>

`Crtp< Type, Parameters...> & arrange (Function &&f=Function(), Indexes &&indexes=Indexes(), const Mask &bitmask=Mask())`

Arrange the container.

- template<class Function = std::less<Type>, class Indexes = Type, class Mask = std::true_type, class Pair = std::pair<Type, typename std::remove_const<typename std::remove_reference<decltype(Vectorizer::get(std::declval<Indexes>(), 0))>::type>::type>, class = typename std::enable_if<(!std::is_base_of<Vectorizer, typename std::decay<Function>::type>::value) && ((std::is_base_of<Vectorizer, Mask>::value) || (std::is_same<std::true_type, Mask>::value))>::type>

`Crtp< Type, Parameters...> sort (Function &&f=Function(), Indexes &&indexes=Indexes(), const Mask &bitmask=Mask()) const`

Sort.

- template<class Function = std::less<Type>, class Indexes = Type, class Mask = std::true_type, class Pair = std::pair<Type, typename std::remove_const<typename std::remove_reference<decltype(Vectorizer::get(std::declval<Indexes>(), 0))>::type>::type>, class = typename std::enable_if<(std::is_base_of<Vectorizer, typename std::decay<Function>::type>::value) && ((std::is_base_of<Vectorizer, Mask>::value) || (std::is_same<std::true_type, Mask>::value))>::type>
 Crtp< Type, Parameters...> & **aarrange** (Function &&f=Function(), Indexes &&indexes=Indexes(), const Mask &bitmask=Mask())
 Absolute arrange the container.
- template<class Function = std::less<Type>, class Indexes = Type, class Mask = std::true_type, class Pair = std::pair<Type, typename std::remove_const<typename std::remove_reference<decltype(Vectorizer::get(std::declval<Indexes>(), 0))>::type>::type>, class = typename std::enable_if<(std::is_base_of<Vectorizer, typename std::decay<Function>::type>::value) && ((std::is_base_of<Vectorizer, Mask>::value) || (std::is_same<std::true_type, Mask>::value))>::type>
 Crtp< Type, Parameters...> **asort** (Function &&f=Function(), Indexes &&indexes=Indexes(), const Mask &bitmask=Mask()) const
 Absolute sort.
- template<class Indexes = Type, class Pair = std::pair<typename std::remove_const<typename std::remove_reference<decltype(Vectorizer::get(std::declval<Indexes>(), 0))>::type>::type, unsigned int>>
 Crtp< Type, Parameters...> & **rearrange** (const Indexes &indexes=Indexes())
 Re-arrange as a function of indexes.
- template<class Indexes = Type, class Pair = std::pair<typename std::remove_const<typename std::remove_reference<decltype(Vectorizer::get(std::declval<Indexes>(), 0))>::type>::type, unsigned int>>
 Crtp< Type, Parameters...> **resort** (const Indexes &indexes=Indexes()) const
 Re-sort as a function of indexes.

Mathematical functions

- template<class Mask = std::true_type, class = typename std::enable_if<(std::is_base_of<Vectorizer, Mask>::value) || (std::is_same<std::true_type, Mask>::value)>::type>
 Crtp< Type, Parameters...> **abs** (const Mask &bitmask=Mask()) const
 Absolute value.
- template<int Exponent = 1>
 Crtp< Type, Parameters...> **pow** () const
 Power metafunction.
- template<class GenericType , class Mask = std::true_type, class = typename std::enable_if<(std::is_base_of<Vectorizer, Mask>::value) || (std::is_same<std::true_type, Mask>::value)>::type>
 Crtp< Type, Parameters...> **pow** (const GenericType &rhs, const Mask &bitmask=Mask()) const
 Power.
- template<int Degree = 1>
 Crtp< Type, Parameters...> **rt** () const
 Root metafunction.
- template<class GenericType , class Mask = std::true_type, class = typename std::enable_if<(std::is_base_of<Vectorizer, Mask>::value) || (std::is_same<std::true_type, Mask>::value)>::type>
 Crtp< Type, Parameters...> **rt** (const GenericType &rhs, const Mask &bitmask=Mask()) const
 Root.
- template<unsigned int Base = 0>
 Crtp< Type, Parameters...> **log** () const
 Log metafunction.
- template<class GenericType , class Mask = std::true_type, class = typename std::enable_if<(std::is_base_of<Vectorizer, Mask>::value) || (std::is_same<std::true_type, Mask>::value)>::type>
 Crtp< Type, Parameters...> **log** (const GenericType &rhs, const Mask &bitmask=Mask()) const
 Logarithm.

Norm

- template<unsigned int Degree = 2, typename NormType = typename std::conditional<std::is_floating_point<Type>::value, Type, double>::type, class Mask = std::true_type, class = typename std::enable_if<(std::is_arithmetic<NormType>::value) && ((std::is_base_of<Vectorizer, Mask>::value) || (std::is_same<std::true_type, Mask>::value))>::type>
 NormType **norm** (const Mask &bitmask=Mask()) const
 Norm.

- template<unsigned int Degree = 2, typename NormType = typename std::conditional<std::is_floating_point<Type>::value, Type, double>::type, class Mask = std::true_type, class = typename std::enable_if<(std::is_arithmetic<NormType>::value) && ((std::is_base_of<Vectorizer, Mask>::value) || (std::is_same<std::true_type, Mask>::value))>::type> Crtp< Type, Parameters...> & renormalize (const Mask &bitmask=Mask())

Renormalize.
- template<unsigned int Degree = 2, typename NormType = typename std::conditional<std::is_floating_point<Type>::value, Type, double>::type, class Mask = std::true_type, class = typename std::enable_if<(std::is_arithmetic<NormType>::value) && ((std::is_base_of<Vectorizer, Mask>::value) || (std::is_same<std::true_type, Mask>::value))>::type> Crtp< Type, Parameters...> normalize (const Mask &bitmask=Mask()) const

Normalize.

Static Public Member Functions

Predefined

- static Crtp< Type, Parameters...> zero ()

Zero array.
- static Crtp< Type, Parameters...> one ()

One array.
- template<class Mask = std::true_type, class = typename std::enable_if<(std::is_base_of<Vectorizer, Mask>::value) || (std::is_same<std::true_type, Mask>::value)>::type>
 static Crtp< Type, Parameters...> value (const Type &source=Type(), const Mask &bitmask=Mask())

Array from value.
- template<class Mask = std::true_type, class = typename std::enable_if<(std::is_base_of<Vectorizer, Mask>::value) || (std::is_same<std::true_type, Mask>::value)>::type>
 static Crtp< Type, Parameters...> random (const Type &lowest=0, const Type &highest=1, const Mask &bitmask=Mask())

Basic random array creation.
- template<class Engine , class Distribution , class Mask = std::true_type, class = typename std::enable_if<(std::decay<Engine>::type::min() != std::decay<Engine>::type::max()) && (!std::is_void<typename std::decay<Distribution>::type::result_type>::value) && ((std::is_base_of<Vectorizer, Mask>::value) || (std::is_same<std::true_type, Mask>::value))>::type>
 static Crtp< Type, Parameters...> random (Engine &&engine, Distribution &&distribution, const Mask &bitmask=Mask())

Generic random array creation.
- template<class Function = unsigned int&& (*)(unsigned int&), class Mask = std::true_type, class = typename std::enable_if<(!std::is_base_of<Vectorizer, typename std::decay<Function>::type>::value) && ((std::is_base_of<Vectorizer, Mask>::value) || (std::is_same<std::true_type, Mask>::value))>::type>
 static Crtp< Type, Parameters...> indexed (Function &&f=std::forward< unsigned int >, const Mask &bitmask=Mask())

Indexed array.
- template<class Function = std::plus<Type>, class Mask = std::true_type, class = typename std::enable_if<(!std::is_base_of<Vectorizer, typename std::decay<Function>::type>::value) && ((std::is_base_of<Vectorizer, Mask>::value) || (std::is_same<std::true_type, Mask>::value))>::type>
 static Crtp< Type, Parameters...> progressive (const Type &first=Type(0), const Type &step=Type(1), Function &&f=Function(), const Mask &bitmask=Mask())

Progressive array.

Test

- static int example ()

Example function.

Public Attributes

- using operator = typedef

Protected Member Functions

Protected lifecycle

- `~AbstractNArray ()`
Protected destructor.

Protected Attributes

Data members

- Type `_data` [Size]
Data contents.

6.9.1 Detailed Description

```
template<typename Kind, unsigned int Size, template<typename, Kind...> class Crtp, typename Type, Kind... Parameters>class
magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >
```

Abstract base class of n-dimensional mathematical arrays.

Provides common base for n-dimensional order 1 mathematical containers thanks to the curiously recurring template pattern (CRTP) trick. The class derives from [StaticVectorizer](#), provides the storage and adds functions specific to arithmetic types. To use it, one has to derive from this class and pass the derived class itself as the CRTP parameter. The derived classes have to satisfy the conditions required by the [Vectorizer](#) base class.

Template Parameters

<code>Kind</code>	Kind of arguments.
<code>Size</code>	Number of elements.
<code>Crtp</code>	Derived CRTP class.
<code>Type</code>	Arithmetic data type.
<code>Parameters</code>	List of parameters.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 template<typename Kind , unsigned int Size, template<typename, Kind...> class Crtp, typename Type , Kind... Parameters> magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::~AbstractNArray()
`[inline], [protected], [default]`

Protected destructor.

Does nothing.

6.9.2.2 template<typename Kind , unsigned int Size, template<typename, Kind...> class Crtp, typename Type , Kind... Parameters> magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::AbstractNArray()
`[inline]`

Implicit empty constructor.

Does nothing.

6.9.2.3 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename FundamentalType , class > **magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::AbstractNArray** (const **AbstractNArray< Kind, Size, Crtp, FundamentalType, Parameters...>** & *source*) [inline]

Implicit conversion constructor.

Provides an implicit conversion from a fundamental type contents.

Template Parameters

<i>FundamentalType</i>	(Fundamental data type.)
------------------------	--------------------------

Parameters

in	<i>source</i>	Source of the copy.
----	---------------	---------------------

6.9.2.4 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename OtherType , class... Misc, class > **magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::AbstractNArray** (const std::initializer_list< OtherType > & *source*, const Misc &... *misc*) [inline]

Implicit initializer list constructor.

Provides an implicit conversion from an initializer list.

Template Parameters

<i>OtherType</i>	(Other data type.)
<i>Misc</i>	(Miscellaneous types.)

Parameters

in	<i>source</i>	Source of the copy.
in	<i>misc</i>	Miscellaneous arguments.

6.9.2.5 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<class... Misc, class > **magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::AbstractNArray** (const Misc &... *misc*) [inline], [explicit]

Explicit generic constructor.

Provides a generic interface to all constructors of the base class. Before calling the associated constructor of the base class, the contents is initialized.

Template Parameters

<i>Misc</i>	(Miscellaneous types.)
-------------	------------------------

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

6.9.3 Member Function Documentation

6.9.3.1 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind...> template<class Function , class Indexes , class Mask , class Pair , class > Crtp< Type, Parameters...> & magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::aarrange(Function && f=Function() , Indexes && indexes = Indexes() , const Mask & bitmask = Mask()) [inline]

Absolute arrange the container.

Sorts the contents by its absolute value and returns a sorted reference using std::sort. A comparison function, a list of indexes and a boolean mask can be passed. The list of indexes is a vectorized container which will be sorted in the same order as the array. If a scalar is passed, nothing is done on it. This is equivalent to the [asort\(\)](#) function except that it works on the contents instead of a copy.

Template Parameters

<i>Function</i>	(Function type : bool (Type, Type).)
<i>Indexes</i>	(Vectorized container type or dummy scalar.)
<i>Mask</i>	(Mask type.)
<i>Pair</i>	(Inner pair type for indexes.)

Parameters

in	<i>f</i>	Function object bool (Type, Type).
in,out	<i>indexes</i>	Vectorized container or dummy scalar to be sorted.
in	<i>bitmask</i>	Boolean mask.

Returns

Self reference.

6.9.3.2 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind...> template<class Mask , class > Crtp< Type, Parameters...> magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::abs(const Mask & bitmask = Mask()) const [inline]

Absolute value.

Applies the std::abs() function to each element.

Template Parameters

<i>Mask</i>	(Mask type.)
-------------	--------------

Parameters

in	<i>bitmask</i>	Boolean mask.
----	----------------	---------------

Returns

Copy.

6.9.3.3 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind...> template<class Mask , class > const Type & magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::amax(const Mask & bitmask = Mask()) const [inline]

Maximum element.

Returns a reference to the absolute maximum element of the container or masked container.

Template Parameters

<i>Mask</i>	(Mask type.)
-------------	--------------

Parameters

<i>in</i>	<i>bitmask</i>	Boolean mask.
-----------	----------------	---------------

Returns

Immutable reference to the element.

Exceptions

<i>std::runtime_error</i>	Empty search.
---------------------------	---------------

6.9.3.4 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<class Mask , class > const Type & magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::amin (const Mask & *bitmask* =Mask ()) const [inline]

Absolute minimum element.

Returns a reference to the absolute minimum element of the container or masked container.

Template Parameters

<i>Mask</i>	(Mask type.)
-------------	--------------

Parameters

<i>in</i>	<i>bitmask</i>	Boolean mask.
-----------	----------------	---------------

Returns

Immutable reference to the element.

Exceptions

<i>std::runtime_error</i>	Empty search.
---------------------------	---------------

6.9.3.5 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename Return , typename First , class... Types, class... Args, class > Crtp< Return, Parameters...> magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::apply (Return(*)(First, Types...) *f*, Args &&... *args*) const [inline]

Application of a function pointer.

Applies a function pointer to each element of the container and returns a copy of the result. For a result *y*, an element *x*, a function *f* and for extra arguments *args*..., an equivalent expression is : *y* = *f*(*x*, *args*...).

Template Parameters

<i>Return</i>	Return type.
<i>First</i>	Type of the first argument of the function.
<i>Types</i>	Extra types of the function.
<i>Args</i>	(Extra types.)

Parameters

in	<i>f</i>	Function pointer <code>Return(First, Types...)</code> .
in	<i>args</i>	Extra arguments of the function.

Returns

Copy.

Warning

In case of implicit cast or extra arguments, types related to the function have to be specified manually as template parameters.

6.9.3.6 `template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type, Kind... Parameters> template<typename Return, typename First, class... Types, class... Args, class Mask, class > Crtp< Return, Parameters...> magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::apply (const Mask & bitmask, Return(*)(First, Types...) f, Args &&... args) const [inline]`

Masked application of a function pointer.

Applies a function pointer to each element of the container where the mask is true and returns a copy of the result. For a result `y`, an element `x`, a function `f` and for extra arguments `args...`, an equivalent expression is : `y = f(x, args...)`.

Template Parameters

<i>Return</i>	Return type.
<i>First</i>	Type of the first argument of the function.
<i>Types</i>	Extra types of the function.
<i>Args</i>	(Extra types.)
<i>Mask</i>	(Mask type.)

Parameters

in	<i>bitmask</i>	Boolean mask.
in	<i>f</i>	Function pointer <code>Return(First, Types...)</code> .
in	<i>args</i>	Extra arguments of the function.

Returns

Copy.

Warning

In case of implicit cast or extra arguments, types related to the function have to be specified manually as template parameters.

6.9.3.7 `template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type, Kind... Parameters> template<class Function, class Indexes, class Mask, class Pair, class > Crtp< Type, Parameters...> & magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::arrange (Function && f = Function(), Indexes && indexes = Indexes(), const Mask & bitmask = Mask()) [inline]`

Arrange the container.

Sorts the contents and returns a sorted reference using `std::sort`. A comparison function, a list of indexes and a boolean mask can be passed. The list of indexes is a vectorized container which will be sorted in the same order as the array. If a scalar is passed, nothing is done on it. This is equivalent to the `sort()` function except that it works on the contents instead of a copy.

Template Parameters

<i>Function</i>	(Function type : <code>bool (Type, Type)</code> .)
<i>Indexes</i>	(Vectorized container type or dummy scalar.)
<i>Mask</i>	(Mask type.)
<i>Pair</i>	(Inner pair type for indexes.)

Parameters

<code>in</code>	<code>f</code>	Function object <code>bool (Type, Type)</code> .
<code>in, out</code>	<code>indexes</code>	Vectorized container or dummy scalar to be sorted.
<code>in</code>	<code>bitmask</code>	Boolean mask.

Returns

Self reference.

```
6.9.3.8 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind...
Parameters> template<class Function , class Indexes , class Mask , class Pair , class > Crtp< Type, Parameters...>
magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::asort( Function && f = Function(),
Indexes && indexes = Indexes(), const Mask & bitmask = Mask() ) const [inline]
```

Absolute sort.

Sorts the contents by its absolute value and returns a sorted copy using `std::sort`. A comparison function, a list of indexes and a boolean mask can be passed. The list of indexes is a vectorized container which will be sorted in the same order as the array. If a scalar is passed, nothing is done on it. This is equivalent to the `aarrange()` function except that it works on a copy instead of the contents.

Template Parameters

<i>Function</i>	(Function type : <code>bool (Type, Type)</code> .)
<i>Indexes</i>	(Vectorized container type or dummy scalar.)
<i>Mask</i>	(Mask type.)
<i>Pair</i>	(Inner pair type for indexes.)

Parameters

<code>in</code>	<code>f</code>	Function object <code>bool (Type, Type)</code> .
<code>in, out</code>	<code>indexes</code>	Vectorized container or dummy scalar to be sorted.
<code>in</code>	<code>bitmask</code>	Boolean mask.

Returns

Copy.

```
6.9.3.9 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind...
Parameters> Type * magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::begin( )
[inline]
```

Iterator to the beginning.

Returns a pointer to the first element.

Returns

Pointer to the beginning.

6.9.3.10 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> const Type * magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::begin () const [inline]

Immutable iterator to the beginning.

Returns a pointer to the first element.

Returns

Immutable pointer to the beginning.

6.9.3.11 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> const Type * magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::cbegin () const [inline]

Forced immutable iterator to the beginning.

Returns a constant pointer to the first element.

Returns

Immutable pointer to the beginning.

6.9.3.12 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> const Type * magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::cend () const [inline]

Forced immutable iterator to the end.

Returns a constant pointer to the position after the last element.

Returns

Immutable pointer to the end.

6.9.3.13 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> std::reverse_iterator< const Type * > magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::crbegin () const [inline]

Forced immutable reverse iterator to the beginning.

Returns a constant reversed pointer to the position after the last element.

Returns

Immutable pointer to the end.

6.9.3.14 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> std::reverse_iterator< const Type * > **magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::crend () const** [inline]

Forced immutable reverse iterator to the end.

Returns a constant reversed pointer to the first element.

Returns

Immutable pointer to the beginning.

6.9.3.15 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> Type * **magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::data ()** [inline]

Direct access to the underlying storage.

Returns a pointer to the first element of the underlying array.

Returns

Pointer to the first element.

6.9.3.16 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> const Type * **magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::data () const** [inline]

Immutable direct access to the underlying storage.

Returns a pointer to the first element of the underlying array.

Returns

Immutable pointer to the first element.

6.9.3.17 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<class Function , class Mask , class > unsigned int **magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::distinct (Function && f = Function (), const Mask & bitmask = Mask ()) const** [inline]

Number of distincts elements.

Counts the number of distincts elements using the provided function with the `std::unique()` algorithm.

Template Parameters

<i>Function</i>	(Function type : <code>bool (Type, Type)</code> .)
<i>Mask</i>	(Mask type.)

Parameters

in	<i>f</i>	Function object <code>bool (Type, Type)</code> .
in	<i>bitmask</i>	Boolean mask.

Returns

Copy of the result.

6.9.3.18 `template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> Type * magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::end () [inline]`

Iterator to the end.

Returns a pointer to the position after the last element.

Returns

Pointer to the end.

6.9.3.19 `template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> const Type * magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::end () const [inline]`

Immutable iterator to the end.

Returns a pointer to the position after the last element.

Returns

Immutable pointer to the end.

6.9.3.20 `template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<class GenericType > bool magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::eq (const GenericType & rhs, const Type & tolerance) const [inline]`

Compare for approximate equality.

Returns true if all elements of the containers are approximately equal, returns false otherwise. The condition is that the difference of all elements of the containers has to be less or equal the absolute value of the tolerance.

Template Parameters

<code>GenericType</code>	(Value or vectorized type.)
--------------------------	-----------------------------

Parameters

<code>in</code>	<code>rhs</code>	Right-hand side.
<code>in</code>	<code>tolerance</code>	Tolerance of approximation.

Returns

Copy of the result.

6.9.3.21 `template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> int magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::example () [static]`

Example function.

Tests and demonstrates the use of [AbstractNArray](#).

Returns

0 if no error.

6.9.3.22 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename IteratorType > unsigned int magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::index (const IteratorType & *it*, typename std::iterator_traits< IteratorType >::iterator_category * = nullptr) const [inline]

Index of an iterator in the array.

Returns the index of the element pointed by an iterator or a pointer.

Template Parameters

<i>IteratorType</i>	(Pointer or iterator type.)
---------------------	-----------------------------

Parameters

in	<i>it</i>	Iterator to the element.
----	-----------	--------------------------

Returns

Copy of the value of the index of the element.

Exceptions

<i>std::out_of_range</i>	Out of range.
--------------------------	---------------

6.9.3.23 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<class Function , class Mask , class > Crtp< Type, Parameters...> magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::indexed (Function && *f* = std::forward<unsigned int>, const Mask & *bitmask* = Mask()) [inline], [static]

Indexed array.

Creates an array based on indexes. The passed function should take an index value and return the associated element value.

Template Parameters

<i>Function</i>	(Function type : Type (unsigned int).)
<i>Mask</i>	(Mask type.)

Parameters

in	<i>f</i>	Function object Type (unsigned int).
in	<i>bitmask</i>	Boolean mask.

Returns

Copy.

6.9.3.24 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<unsigned int Base> Crtp< Type, Parameters...> magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::log () const [inline]

Log metafunction.

Computes the logarithm with compile-time options. If the base is 0, then the natural base logarithm is called. If the data type is a floating point, and if the base is 2 or 10, then the `std::log2()` or `std::log10` function is called. For other bases, the division implying the logarithm of the base is used. If the data type is not a floating point, the exact integral part of the logarithm in the specified base is computed using loops.

Template Parameters

<code>Base</code>	Logarithm base.
-------------------	-----------------

Returns

Copy.

Exceptions

<code>std::domain_error</code>	Logarithm of a negative integer undefined.
--------------------------------	--

6.9.3.25 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<class GenericType , class Mask , class > Crtp< Type, Parameters...> magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::log (const GenericType & rhs, const Mask & bitmask = Mask ()) const [inline]

Logarithm.

Computes the logarithm in the specified base of each element using the `std::log()` function.

Template Parameters

<code>GenericType</code>	(Value or vectorized type.)
<code>Mask</code>	(Mask type.)

Parameters

in	<code>rhs</code>	Right-hand side.
in	<code>bitmask</code>	Boolean mask.

Returns

Copy.

6.9.3.26 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename Return , class Coefficient , class Mask , class > Return magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::mean (const Coefficient & coefficient = Coefficient (1), const Mask & bitmask = Mask ()) const [inline]

Mean value.

Computes the mean value of the container or masked container. If a coefficient container is provided, the weighted mean is computed : $\bar{x} = \frac{\sum_{i=1}^n w_i x_i}{\sum_{i=1}^n w_i}$.

Template Parameters

<i>Return</i>	Return type.
<i>Coefficient</i>	(Coefficient type.)
<i>Mask</i>	(Mask type.)

Parameters

in	<i>coefficient</i>	Coefficients of weighted mean.
in	<i>bitmask</i>	Boolean mask.

Returns

Copy of the mean value.

6.9.3.27 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename Return , typename First , class... Types, class... Args, class > Crtp< Type, Parameters...> & magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::modify (Return(*)(First, Types...) f, Args &&... args) [inline]

Modification by a function pointer.

Modifies the container by applying a function pointer to each element. For an element *x*, a function *f* and for extra arguments *args...*, an equivalent expression is : *x* = *f* (*x*, *args...*). The return type is used as an internal cast before affectation.

Template Parameters

<i>Return</i>	Return type.
<i>First</i>	Type of the first argument of the function.
<i>Types</i>	Extra types of the function.
<i>Args</i>	(Extra types.)

Parameters

in	<i>f</i>	Function pointer Return(First, Types...).
in	<i>args</i>	Extra arguments of the function.

Returns

Self reference.

Warning

In case of implicit cast or extra arguments, types related to the function have to be specified manually as template parameters.

6.9.3.28 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename Return , typename First , class... Types, class... Args, class Mask , class > Crtp< Type, Parameters...> & magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::modify (const Mask & *bitmask*, Return(*)(First, Types...) f, Args &&... args) [inline]

Masked modification by a function pointer.

Modifies the container by applying a function pointer to each element where the mask is true. For an element *x*, a function *f* and for extra arguments *args...*, an equivalent expression is : *x* = *f* (*x*, *args...*). The return type is used as an internal cast before affectation.

Template Parameters

<i>Return</i>	Return type.
<i>First</i>	Type of the first argument of the function.
<i>Types</i>	Extra types of the function.
<i>Args</i>	(Extra types.)
<i>Mask</i>	(Mask type.)

Parameters

in	<i>bitmask</i>	Boolean mask.
in	<i>f</i>	Function pointer <code>Return(First, Types...)</code> .
in	<i>args</i>	Extra arguments of the function.

Returns

Self reference.

Warning

In case of implicit cast or extra arguments, types related to the function have to be specified manually as template parameters.

6.9.3.29 `template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type, Kind... Parameters> template<class GenericType > bool magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::ne (const GenericType & rhs, const Type & tolerance) const [inline]`

Compare for noticeable difference.

Returns true if at least one element is noticeably different in the two containers, returns false otherwise. The condition is that at least one difference of two elements of the containers has to be strictly greater than the absolute value of the tolerance.

Template Parameters

<i>GenericType</i>	(Value or vectorized type.)
--------------------	-----------------------------

Parameters

in	<i>rhs</i>	Right-hand side.
in	<i>tolerance</i>	Tolerance of approximation.

Returns

Copy of the result.

6.9.3.30 `template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type, Kind... Parameters> template<unsigned int Degree, typename NormType , class Mask , class > NormType magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::norm (const Mask & bitmask = Mask ()) const [inline]`

Norm.

Computes the p-norm of the array : $\|x\|_p = (\sum_{i=1}^n |x_i|^p)^{1/p}$ where p is specified by the `NormType` template parameter. If it is equal to zero, then the infinite norm is taken.

Template Parameters

<i>Degree</i>	Norm degree.
<i>NormType</i>	Norm type.

Parameters

<i>in</i>	<i>bitmask</i>	Boolean mask.
-----------	----------------	---------------

Returns

Copy of the norm.

6.9.3.31 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters>template<unsigned int Degree, typename NormType , class Mask , class > Crtp< Type, Parameters...> magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::normalize (const Mask & *bitmask* = Mask ()) const [inline]

Normalize.

Normalizes the contents using the the p-norm of the array : $\|x\|_p = (\sum_{i=1}^n |x_i|^p)^{1/p}$ where p is specified by the *NormType* template parameter. If it is equal to zero, then the infinite norm is taken. This is equivalent to the [renormalize\(\)](#) function except that it works on a copy instead of the contents.

Template Parameters

<i>Degree</i>	Norm degree.
<i>NormType</i>	Norm type.

Parameters

<i>in</i>	<i>bitmask</i>	Boolean mask.
-----------	----------------	---------------

Returns

Self reference.

Exceptions

<i>std::domain_error</i>	The norm is not normal.
<i>std::domain_error</i>	The norm is not normal.

6.9.3.32 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> bool magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::null (const Type & *tolerance*) const [inline]

Check whether all elements are approximately null.

Returns true if all elements are approximately set to their default value, returns false otherwise. The condition is that their absolute value is less or equal the absolute value of the tolerance.

Parameters

<i>in</i>	<i>tolerance</i>	Tolerance of approximation.
-----------	------------------	-----------------------------

Returns

Copy of the result of the test.

6.9.3.33 `template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> Crtp< Type, Parameters...> magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::one() [inline], [static]`

One array.

Creates an array filled with ones.

Returns

Copy.

6.9.3.34 `template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> Type & magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::operator[](const unsigned int i) [inline]`

Direct access to the element.

Provides a direct access to the specified element.

Parameters

in	<i>i</i>	Index of the element.
----	----------	-----------------------

Returns

Reference to the element.

6.9.3.35 `template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> const Type & magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::operator[](const unsigned int i) const [inline]`

Immutable direct access to the element.

Provides a constant direct access to the specified element.

Parameters

in	<i>i</i>	Index of the element.
----	----------	-----------------------

Returns

Immutable reference to the element.

6.9.3.36 `template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<int Exponent> Crtp< Type, Parameters...> magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::pow() const [inline]`

Power metafunction.

Recursively calls the multiplication operator at compile-time computing the integer exponentiation of the array. Negative exponent inverses the result.

Template Parameters

<i>Exponent</i>	Exponent of the power function.
-----------------	---------------------------------

Returns

Copy.

```
6.9.3.37 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type
, Kind... Parameters> template<class GenericType , class Mask , class > Crtp< Type, Parameters...>
magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::pow ( const GenericType & rhs, const Mask
& bitmask =Mask () ) const [inline]
```

Power.

Computes the power for the specified exponent of each element using the `std::pow()` function.

Template Parameters

<i>GenericType</i>	(Value or vectorized type.)
<i>Mask</i>	(Mask type.)

Parameters

in	<i>rhs</i>	Right-hand side.
in	<i>bitmask</i>	Boolean mask.

Returns

Copy.

```
6.9.3.38 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type,
Kind... Parameters> template<class Function , class Mask , class > Crtp< Type, Parameters...>
magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::progressive ( const Type & init =
Type(0), const Type & step =Type(1), Function && f =Function (), const Mask & bitmask =Mask () )
[inline], [static]
```

Progressive array.

Creates an array filled with progressive values. The next non-masked element is equal to the result of the function applied to the previous non-masked element and the step.

Template Parameters

<i>Function</i>	(Function type : <code>Type(Type, Type)</code> .)
<i>Mask</i>	(Mask type.)

Parameters

in	<i>init</i>	Initial value.
in	<i>step</i>	Step of progression.
in	<i>f</i>	Function object <code>Type(Type, Type)</code> .
in	<i>bitmask</i>	Boolean mask.

Returns

Copy.

6.9.3.39 `template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type, Kind... Parameters> template<class Mask , class > Crtp< Type, Parameters...> magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::random (const Type & minimum = 0, const Type & maximum = 1, const Mask & bitmask = Mask()) [inline], [static]`

Basic random array creation.

Creates a random array filled with random values generated with a uniform distribution over the provided interval.

Template Parameters

<i>Mask</i>	(Mask type.)
-------------	--------------

Parameters

in	<i>minimum</i>	Minimum value of the distribution.
in	<i>maximum</i>	Maximum value of the distribution.
in	<i>bitmask</i>	Boolean mask.

Returns

Copy.

Warning

As the internal engine is a static one, do not use this function in parallel.

6.9.3.40 `template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type, Kind... Parameters> template<class Engine , class Distribution , class Mask , class > Crtp< Type, Parameters...> magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::random (Engine && engine, Distribution && distribution, const Mask & bitmask = Mask()) [inline], [static]`

Generic random array creation.

Creates a random array filled with random values generated from the provided distribution and engine.

Template Parameters

<i>Engine</i>	(Random engine type.)
<i>Distribution</i>	(Random distribution type.)
<i>Mask</i>	(Mask type.)

Parameters

in,out	<i>engine</i>	Random engine.
in,out	<i>distribution</i>	Random distribution.
in	<i>bitmask</i>	Boolean mask.

Returns

Copy.

6.9.3.41 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> std::reverse_iterator< Type * > **magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::rbegin ()** [inline]

Reverse iterator to the beginning.

Returns a reversed pointer to the position after the last element.

Returns

Pointer to the end.

6.9.3.42 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> std::reverse_iterator< const Type * > **magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::begin () const** [inline]

Immutable reverse iterator to the beginning.

Returns a reversed pointer to the position after the last element.

Returns

Immutable pointer to the end.

6.9.3.43 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<class Indexes , class Pair > Crtp< Type, Parameters...> & **magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::rearrange (const Indexes & indexes = Indexes())** [inline]

Re-arrange as a function of indexes.

Sorts the contents by the provided indexes previously generated by an [arrange\(\)](#) or a [sort\(\)](#) function. It can be used to sort several arrays in the same order of a single one. If no indexes or a scalar is provided, the container is randomly shuffled. This is equivalent to the [resort\(\)](#) function except that it works on the contents instead of a copy.

Template Parameters

<i>Indexes</i>	(Vectorized container type or dummy scalar.)
<i>Pair</i>	(Inner pair type for indexes.)

Parameters

in	<i>indexes</i>	Vectorized container of indexes.
----	----------------	----------------------------------

Returns

Self reference.

6.9.3.44 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> std::reverse_iterator< Type * > **magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::rend ()** [inline]

Reverse iterator to the end.

Returns a reversed pointer to the first element.

Returns

Pointer to the beginning.

6.9.3.45 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> std::reverse_iterator< const Type * > magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::rend() const [inline]

Immutable reverse iterator to the end.

Returns a reversed pointer to the first element.

Returns

Immutable pointer to the beginning.

6.9.3.46 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<unsigned int Degree, typename NormType , class Mask , class > Crtp< Type, Parameters...> & magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::renormalize(const Mask & bitmask = Mask()) [inline]

Renormalize.

Renormalizes the contents using the the p-norm of the array : $\|x\|_p = (\sum_{i=1}^n |x_i|^p)^{1/p}$ where p is specified by the NormType template parameter. If it is equal to zero, then the infinite norm is taken. This is equivalent to the [normalize\(\)](#) function except that it works on the contents instead of a copy.

Template Parameters

<i>Degree</i>	Norm degree.
<i>NormType</i>	Norm type.

Parameters

<i>in</i>	<i>bitmask</i>	Boolean mask.
-----------	----------------	---------------

Returns

Self reference.

Exceptions

<i>std::domain_error</i>	The norm is not normal.
<i>std::domain_error</i>	The norm is not normal.

6.9.3.47 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<class Indexes , class Pair > Crtp< Type, Parameters...> magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::resort(const Indexes & indexes = Indexes()) const [inline]

Re-sort as a function of indexes.

Sorts a copy by the provided indexes previously generated by an [arrange\(\)](#) or a [sort\(\)](#) function. It can be used to sort several arrays in the same order of a single one. If no indexes or a scalar is provided, the container is randomly shuffled. This is equivalent to the [rearrange\(\)](#) function except that it works on a copy instead of the contents.

Template Parameters

<i>Indexes</i>	(Vectorized container type or dummy scalar.)
<i>Pair</i>	(Inner pair type for indexes.)

Parameters

in	<i>indexes</i>	Vectorized container of indexes.
----	----------------	----------------------------------

Returns

Copy.

6.9.3.48 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<int Degree> Crtp< Type, Parameters...> magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::rt() const [inline]

Root metafunction.

Calls the inverse of the power function for each element at compile-time. For 2 and 3 exponents, the `std::sqrt()` and `std::cbrt()` functions are called, otherwise `std::pow()` with the inverse exponent is called. At the end, the result is casted to the array type.

Template Parameters

<i>Degree</i>	Root degree.
---------------	--------------

Returns

Copy.

6.9.3.49 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<class GenericType , class Mask , class > Crtp< Type, Parameters...> magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::rt(const GenericType & rhs, const Mask & bitmask = Mask()) const [inline]

Root.

Computes the root for the specified degree of each element using the `std::pow()` function.

Template Parameters

<i>GenericType</i>	(Value or vectorized type.)
<i>Mask</i>	(Mask type.)

Parameters

in	<i>rhs</i>	Right-hand side.
in	<i>bitmask</i>	Boolean mask.

Returns

Copy.

6.9.3.50 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind...> template<typename Return , typename Correction , class Coefficient , class Mask , class > Return magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::sigma (const Correction & *correction* = Correction(0), const Coefficient & *coefficient* = Coefficient(1), const Mask & *bitmask* = Mask()) const [inline]

Standard deviation.

Computes the standard deviation of the container or masked container. The correction, oftenly equals to 0 or -1 , allows to correct the standard deviation from the bias. If a coefficient container is provided, the standard deviation of the weighted mean is computed : $\sigma = \sqrt{\frac{\sum_{i=1}^n w_i(x_i - \bar{x})^2}{\sum_{i=1}^n w_i + c}}$.

Template Parameters

<i>Return</i>	Return type.
<i>Correction</i>	(Correction arithmetic type.)
<i>Coefficient</i>	(Coefficient type.)
<i>Mask</i>	(Mask type.)

Parameters

in	<i>correction</i>	Additive term at denominator for bias control.
in	<i>coefficient</i>	Coefficients of weighted mean.
in	<i>bitmask</i>	Boolean mask.

Returns

Copy of the mean value.

6.9.3.51 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind...> template<class Function , class Indexes , class Mask , class Pair , class > Crtp< Type, Parameters...> magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::sort (Function && *f* = Function(), Indexes && *indexes* = Indexes(), const Mask & *bitmask* = Mask()) const [inline]

Sort.

Sorts the contents and returns a sorted copy using `std::sort`. A comparison function, a list of indexes and a boolean mask can be passed. The list of indexes is a vectorized container which will be sorted in the same order as the array. If a scalar is passed, nothing is done on it. This is equivalent to the `arrange()` function except that it works on a copy instead of the contents.

Template Parameters

<i>Function</i>	(Function type : <code>bool (Type, Type)</code> .)
<i>Indexes</i>	<code>Vectorized</code> container type or dummy scalar.)
<i>Mask</i>	(Mask type.)
<i>Pair</i>	(Inner pair type for indexes.)

Parameters

in	<i>f</i>	Function object <code>bool (Type, Type)</code> .
in,out	<i>indexes</i>	<code>Vectorized</code> container or dummy scalar to be sorted.
in	<i>bitmask</i>	Boolean mask.

Returns

Copy.

6.9.3.52 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<class Function , class Mask , class > bool magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::unicity (Function && f = Function (), const Mask & bitmask = Mask ()) const [inline]

Unicity of elements.

Checks for unicity of elements using the provided function with the `std::unique()` algorithm. If two elements are equal, the function returns false, and true otherwise.

Template Parameters

<i>Function</i>	(Function type : <code>bool (Type, Type)</code> .)
<i>Mask</i>	(Mask type.)

Parameters

in	<i>f</i>	Function object <code>bool (Type, Type)</code> .
in	<i>bitmask</i>	Boolean mask.

Returns

Copy of the result.

6.9.3.53 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type, Kind... Parameters> template<class Mask , class > Crtp< Type, Parameters...> magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::value (const Type & source = Type (), const Mask & bitmask = Mask ()) [inline], [static]

Array from value.

Creates an array filled with values.

Template Parameters

<i>Mask</i>	(Mask type.)
-------------	--------------

Parameters

in	<i>source</i>	Source value.
in	<i>bitmask</i>	Boolean mask.

Returns

Copy.

6.9.3.54 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> Crtp< Type, Parameters...> magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::zero () [inline], [static]

Zero array.

Creates an array filled with zeroes.

Returns

Copy.

6.9.4 Member Data Documentation

6.9.4.1 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type, Kind... Parameters> Type magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::_data[Size]
[protected]

Data contents.

6.9.4.2 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type, Kind... Parameters> using magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >::operator =

The documentation for this class was generated from the following file:

- /data/home/mbreton/magrathea/pathfinder/src/magrathea/abstractnarray.h

6.10 `magrathea::AbstractShape` Class Reference

Common abstraction of n-dimensional shapes.

```
#include <abstractshape.h>
```

Inheritance diagram for magrathea::AbstractShape:



Static Public Member Functions

Mathematical functions

- template<int Exponent, typename Type >
static constexpr Type **pow** (const Type &value)
Integral exponentiation.
 - template<int Root, typename Type >
static constexpr Type **rt** (const Type &value)
Floating-point root.
 - template<int Value, typename Type = long long int>
static constexpr Type **factorial** ()
Factorial.
 - template<int Value, bool Odd, typename Type = long long int>
static constexpr Type **factorial** ()
 - template<int Set, int Subset, typename Type = long long int>
static constexpr Type **binomial** ()
Binomial.
 - template<int Set, int Subset, bool Repetition, typename Type = long long int>
static constexpr Type **combinations** ()
Combinations.
 - template<int Set, int Subset, bool Repetition, typename Type = long long int>
static constexpr Type **variations** ()
Variations.

- template<long long int Value, long long int Even = 1, long long int Odd = -1, typename Type = long long int>
static constexpr Type **alt** ()
Alternating sequence.

Constants

- template<int Exponent = 1, typename Type = double>
static constexpr Type **golden** ()
Golden ratio.
- template<int Exponent = 1, typename Type = double>
static constexpr Type **pi** ()
Pi.
- template<int Exponent = 1, typename Type = double>
static constexpr Type **sqrtpi** ()
Square root of pi.

Test

- static int **example** ()
Example function.

Protected Member Functions

Protected lifecycle

- **~AbstractShape** ()
Protected destructor.

Lifecycle

- template<class... Dummy>
AbstractShape (const Dummy &...dummy)
Explicit generic constructor.

6.10.1 Detailed Description

Common abstraction of n-dimensional shapes.

Provides compile-time helper functions to deal with n-dimensional geometrical objects like hypercubes or hyperspheres.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 **magrathea::AbstractShape::~AbstractShape()** [inline], [protected], [default]

Protected destructor.

Avoids direct instantiation of the class, and only allows it through its derived children.

6.10.2.2 template<class... Dummy> **magrathea::AbstractShape::AbstractShape(const Dummy &... dummy)** [inline], [explicit], [protected]

Explicit generic constructor.

Provides a constructor with dummy parameters for hackery purposes.

Template Parameters

Dummy	(Dummy types.)
-------	----------------

Parameters

in	dummy	Dummy arguments.
----	-------	------------------

6.10.3 Member Function Documentation

6.10.3.1 `template<long long int Value, long long int Even, long long int Odd, typename Type > constexpr Type magrathea::AbstractShape::alt() [static]`

Alternating sequence.

Computes the specified value of an alternating sequence.

Template Parameters

<i>Value</i>	Value of the argument n .
<i>Even</i>	Value if even n_0 .
<i>Odd</i>	Value if odd n_1 .
<i>Type</i>	(Result type.)

Returns

n_0 if odd, n_1 if even.

6.10.3.2 `template<int Set, int Subset, typename Type > constexpr Type magrathea::AbstractShape::binomial() [static]`

Binomial.

Computes the binomial coefficient of the specified integers, including extension to negative numbers.

Template Parameters

<i>Set</i>	Value of the set n .
<i>Subset</i>	Value of the subset k .
<i>Type</i>	(Result type.)

Returns

nk .

6.10.3.3 `template<int Set, int Subset, bool Repetition, typename Type > constexpr Type magrathea::AbstractShape::combinations() [static]`

Combinations.

Computes the number of combinations with or without repetitions for the specified integers.

Template Parameters

<i>Set</i>	Value of the set n .
<i>Subset</i>	Value of the subset k .
<i>Repetition</i>	Repetition specifier.
<i>Type</i>	(Result type.)

Returns
 C_n^k or ${}^R C_n^k$.
6.10.3.4 int magrathea::AbstractShape::example () [static]

Example function.

Tests and demonstrates the use of [AbstractShape](#).

Returns

0 if no error.

6.10.3.5 template<int Value, bool Odd, typename Type > constexpr Type magrathea::AbstractShape::factorial () [static]

Factorial.

Computes the factorial of the specified integer.

Template Parameters

<i>Value</i>	Value of the argument n .
<i>Type</i>	(Result type.)

Returns
 $n!$.

Computes the double factorial of the specified integer: multiplies the number by all inferior numbers that are even or odd depending on the parameter.

Template Parameters

<i>Value</i>	Value of the argument n .
<i>Odd</i>	Odd double factorial if true, even otherwise.
<i>Type</i>	(Result type.)

Returns
 $n!!$.
6.10.3.6 template<int Value, bool Odd, typename Type = long long int> static constexpr Type magrathea::AbstractShape::factorial () [static]**6.10.3.7 template<int Exponent, typename Type > constexpr Type magrathea::AbstractShape::golden () [static]**

Golden ratio.

Computes the value of $\varphi = \frac{1+\sqrt{5}}{2}$ at the given power.

Template Parameters

<i>Exponent</i>	Value of the exponent n .
<i>Type</i>	(Result type.)

Returns
 $\varphi^n.$
6.10.3.8 template<int Exponent, typename Type > constexpr Type magrathea::AbstractShape::pi() [static]

Pi.

Computes the value of π at the given power.**Template Parameters**

<i>Exponent</i>	Value of the exponent n .
<i>Type</i>	(Result type.)

Returns
 $\pi^n.$
6.10.3.9 template<int Exponent, typename Type > constexpr Type magrathea::AbstractShape::pow(const Type & value) [static]

Integral exponentiation.

Computes the integral exponentiation of the specified value.

Template Parameters

<i>Exponent</i>	Value of the exponent n .
<i>Type</i>	(Result type.)

Parameters

<i>in</i>	<i>value</i>	Value of the argument x .
-----------	--------------	-----------------------------

Returns
 $x^n.$
6.10.3.10 template<int Degree, typename Type > constexpr Type magrathea::AbstractShape::rt(const Type & value) [static]

Floating-point root.

Computes the floating-point root of the specified value.

Template Parameters

<i>Degree</i>	Value of the degree n .
<i>Type</i>	(Result type.)

Parameters

<i>in</i>	<i>value</i>	Value of the argument x .
-----------	--------------	-----------------------------

Returns

$$\sqrt[n]{x}.$$

6.10.3.11 template<int Exponent, typename Type > constexpr Type magrathea::AbstractShape::sqrtPi() [static]

Square root of pi.

Computes the value of the square root of π at the given power.

Template Parameters

<i>Exponent</i>	Value of the exponent n .
<i>Type</i>	(Result type.)

Returns

$$(\sqrt{\pi})^n.$$

6.10.3.12 template<int Set, int Subset, bool Repetition, typename Type > constexpr Type magrathea::AbstractShape::variations() [static]

Variations.

Computes the number of variations with or without repetitions for the specified integers.

Template Parameters

<i>Set</i>	Value of the set n .
<i>Subset</i>	Value of the subset k .
<i>Repetition</i>	Repetition specifier.
<i>Type</i>	(Result type.)

Returns

$$V_n^k \text{ or } {}^R V_n^k.$$

The documentation for this class was generated from the following file:

- /data/home/mbreton/magrathea_pathfinder/src/magrathea/abstractshape.h

6.11 magrathea::AbstractStep< Crtp, Scalar, Array, Tuple > Class Template Reference

Abstraction of an evolution step.

```
#include <abstractstep.h>
```

Public Member Functions

Lifecycle

- [AbstractStep\(\)](#)
Implicit empty constructor.
- template<class OtherCrtp , class OtherScalar , class OtherArray , class OtherTuple >
[AbstractStep](#) (const [AbstractStep](#)< OtherCrtp, OtherScalar, OtherArray, OtherTuple > &source)

Explicit conversion constructor.

- template<class OtherScalar , class OtherArray = Array, class OtherTuple = Tuple, class = typename std::enable_if<(std::is_constructible<Scalar, OtherScalar>::value) && (std::is_constructible<Array, OtherArray>::value) && (std::is_constructible<Tuple, OtherTuple>::value)>::type>

AbstractStep (const OtherScalar &scalar, const OtherArray &array=OtherArray(), const OtherTuple &tuple=OtherTuple())

Explicit step constructor.

Operators

- Crtp & **operator=** (const **AbstractStep**< Crtp, Scalar, Array, Tuple > &rhs)
Copy assignment operator.
- template<class OtherCrtp , class OtherScalar , class OtherArray , class OtherTuple >
Crtp & **operator=** (const **AbstractStep**< OtherCrtp, OtherScalar, OtherArray, OtherTuple > &rhs)
Conversion assignment operator.
- template<class OtherCrtp , class OtherScalar , class OtherArray , class OtherTuple >
bool **operator==** (const **AbstractStep**< OtherCrtp, OtherScalar, OtherArray, OtherTuple > &rhs) const
Equal to.
- template<class OtherCrtp , class OtherScalar , class OtherArray , class OtherTuple >
bool **operator!=** (const **AbstractStep**< OtherCrtp, OtherScalar, OtherArray, OtherTuple > &rhs) const
Not equal to.

Assignment

- Crtp & **assign** ()
Empty assignment.
- Crtp & **assign** (const **AbstractStep**< Crtp, Scalar, Array, Tuple > &source)
Copy assignment.
- template<class OtherCrtp , class OtherScalar , class OtherArray , class OtherTuple >
Crtp & **assign** (const **AbstractStep**< OtherCrtp, OtherScalar, OtherArray, OtherTuple > &source)
Conversion assignment.
- template<class OtherScalar , class OtherArray = Array, class OtherTuple = Tuple, class = typename std::enable_if<(std::is_constructible<Scalar, OtherScalar>::value) && (std::is_constructible<Array, OtherArray>::value) && (std::is_constructible<Tuple, OtherTuple>::value)>::type>

Crtp & **assign** (const OtherScalar &scalar, const OtherArray &array=OtherArray(), const OtherTuple &tuple=OtherTuple())

Step assignment.

Management

- Crtp & **nullify** ()
Nullify.
- Crtp **copy** () const
Copy.
- template<class OtherCrtp = Crtp, class = typename std::enable_if<std::is_constructible<OtherCrtp, Crtp>::value>::type>
OtherCrtp **cast** () const
Cast.

Data

- template<class... Dummy, class Type = typename std::conditional<sizeof...(Dummy) == 0, std::tuple<Scalar, Array, Tuple>, void>-::type, class = typename std::enable_if<sizeof...(Dummy) == 0>::type, class = typename std::enable_if<std::is_convertible<Type, typename std::conditional<sizeof...(Dummy) == 0, std::tuple<Scalar, Array, Tuple>, void>::type>::value>::type>

Type & **data** (Dummy...)

Unified data access.

- template<class... Dummy, class Type = typename std::conditional<sizeof...(Dummy) == 0, std::tuple<Scalar, Array, Tuple>, void>-::type, class = typename std::enable_if<sizeof...(Dummy) == 0>::type, class = typename std::enable_if<std::is_convertible<Type, typename std::conditional<sizeof...(Dummy) == 0, std::tuple<Scalar, Array, Tuple>, void>::type>::value>::type>

const Type & **data** (Dummy...) const

Unified data getter.

- template<class Type , class = typename std::enable_if<std::is_convertible<Type, typename std::conditional<!std::is_void<Type>->::value, std::tuple<Scalar, Array, Tuple>, void>::type>::value>::type>
 Crtp & **data** (const Type &value)

Unified data setter.

Protected Member Functions

Protected lifecycle

- **~AbstractStep ()**
Protected destructor.

6.11.1 Detailed Description

template<class Crtp, class Scalar, class Array, class Tuple>class magrathea::AbstractStep< Crtp, Scalar, Array, Tuple >

Abstraction of an evolution step.

This class is an abstraction of an evolution step with common functions, like assignment, copy, getters and setters. A step object is basically a standard structure with additional features. The internal behaviour is based on three groups of quantities :

- id : a scalar id of the step
- core : an array of basic quantities
- extra : a tuple of derived quantities

Template Parameters

<i>Crtp</i>	Derived CRTP class.
<i>Scalar</i>	Scalar type of id.
<i>Array</i>	Array type of core quantities.
<i>Tuple</i>	Tuple type of extra quantities.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 template<class Crtp , class Scalar , class Array , class Tuple > **magrathea::AbstractStep< Crtp, Scalar, Array, Tuple >::~AbstractStep ()** [inline], [protected], [default]

Protected destructor.

Avoids direct instantiation of the class, and only allows it through its derived children.

6.11.2.2 template<class Crtp , class Scalar , class Array , class Tuple > **magrathea::AbstractStep< Crtp, Scalar, Array, Tuple >::AbstractStep ()** [inline]

Implicit empty constructor.

Provides an implicit construction of an object initialized to its default value.

6.11.2.3 template<class Crtp , class Scalar , class Array , class Tuple > template<class OtherCrtp , class OtherScalar , class OtherArray , class OtherTuple > **magrathea::AbstractStep< Crtp, Scalar, Array, Tuple >::AbstractStep** (const **AbstractStep< OtherCrtp, OtherScalar, OtherArray, OtherTuple > & source**) [inline], [explicit]

Explicit conversion constructor.

Provides an explicit construction from another type of object.

Template Parameters

<i>OtherCrtp</i>	(Other derived CRTP class.)
<i>OtherScalar</i>	(Other scalar type of id.)
<i>OtherArray</i>	(Other array type of core quantities.)
<i>OtherTuple</i>	(Other tuple type of extra quantities.)

Parameters

in	<i>source</i>	Source of the copy.
----	---------------	---------------------

6.11.2.4 template<class Crtp , class Scalar , class Array , class Tuple > template<class OtherScalar , class OtherArray , class OtherTuple , class > **magrathea::AbstractStep< Crtp, Scalar, Array, Tuple >::AbstractStep** (const *OtherScalar & scalar*, const *OtherArray & array* = *OtherArray()*, const *OtherTuple & tuple* = *OtherTuple()*) [inline], [explicit]

Explicit step constructor.

Provides an explicit construction from step components.

Template Parameters

<i>OtherScalar</i>	(Other scalar type of id.)
<i>OtherArray</i>	(Other array type of core quantities.)
<i>OtherTuple</i>	(Other tuple type of extra quantities.)

Parameters

in	<i>scalar</i>	Source of the scalar.
in	<i>array</i>	Source of the array.
in	<i>tuple</i>	Source of the tuple.

6.11.3 Member Function Documentation

6.11.3.1 template<class Crtp , class Scalar , class Array , class Tuple > **Crtp & magrathea::AbstractStep< Crtp, Scalar, Array, Tuple >::assign()** [inline]

Empty assignment.

Assigns contents from an object initialized to its default value.

Returns

Self reference.

6.11.3.2 template<class Crtp, class Scalar, class Array, class Tuple> **Crtp & magrathea::AbstractStep< Crtp, Scalar, Array, Tuple >::assign (const AbstractStep< Crtp, Scalar, Array, Tuple > & source)** [inline]

Copy assignment.

Assigns contents from the same type of object.

Parameters

in	<i>source</i>	Source of the copy.
----	---------------	---------------------

Returns

Self reference.

6.11.3.3 template<class Crtp , class Scalar , class Array , class Tuple > template<class OtherCrtp , class OtherScalar , class OtherArray , class OtherTuple > Crtp & magrathea::AbstractStep< Crtp, Scalar, Array, Tuple >::assign (const AbstractStep< OtherCrtp, OtherScalar, OtherArray, OtherTuple > & *source*) [inline]

Conversion assignment.

Assigns contents from another type of object.

Template Parameters

<i>OtherCrtp</i>	(Other derived CRTP class.)
<i>OtherScalar</i>	(Other scalar type of id.)
<i>OtherArray</i>	(Other array type of core quantities.)
<i>OtherTuple</i>	(Other tuple type of extra quantities.)

Parameters

in	<i>source</i>	Source of the copy.
----	---------------	---------------------

Returns

Self reference.

6.11.3.4 template<class Crtp , class Scalar , class Array , class Tuple > template<class OtherScalar , class OtherArray , class OtherTuple , class > Crtp & magrathea::AbstractStep< Crtp, Scalar, Array, Tuple >::assign (const OtherScalar & *scalar*, const OtherArray & *array* = OtherArray(), const OtherTuple & *tuple* = OtherTuple()) [inline]

[Step](#) assignment.

Assigns contents from step components.

Template Parameters

<i>OtherScalar</i>	(Other scalar type of id.)
<i>OtherArray</i>	(Other array type of core quantities.)
<i>OtherTuple</i>	(Other tuple type of extra quantities.)

Parameters

in	<i>scalar</i>	Source of the scalar.
in	<i>array</i>	Source of the array.
in	<i>tuple</i>	Source of the tuple.

Returns

Self reference.

**6.11.3.5 template<class Crtp , class Scalar , class Array , class Tuple > template<class OtherCrtp , class > OtherCrtp
magrathea::AbstractStep< Crtp, Scalar, Array, Tuple >::cast () const [inline]**

Cast.

Casts contents to another object type.

Template Parameters

<i>OtherCrtp</i>	Other derived CRTP class.
------------------	---------------------------

Returns

Casted copy.

**6.11.3.6 template<class Crtp , class Scalar , class Array , class Tuple > Crtp magrathea::AbstractStep< Crtp, Scalar,
Array, Tuple >::copy () const [inline]**

Copy.

Generates a copy of the object.

Returns

Copy.

**6.11.3.7 template<class Crtp , class Scalar , class Array , class Tuple > template<class... Dummy, class Type , class , class >
Type & magrathea::AbstractStep< Crtp, Scalar, Array, Tuple >::data (Dummy...) [inline]**

Unified data access.

Unified data inner component access.

Unified data component access.

Provides a direct access to the data.

Template Parameters

<i>Dummy</i>	(Dummy types.)
<i>Type</i>	(Data std::tuple<Scalar, Array, Tuple> type.)

Returns

Reference to the data.

Provides a direct access to the specified component of the data.

Template Parameters

<i>Index</i>	Index of the component.
<i>Dummy</i>	(Dummy types.)
<i>Type</i>	(Component type.)

Returns

Reference to the component of the data.

Provides a direct access to the specified inner component of the specified component of the data.

Template Parameters

<i>Index</i>	Index of the component.
<i>Subscript</i>	Subscript of the inner component.
<i>Dummy</i>	(Dummy types.)
<i>Type</i>	(Inner component type.)

Returns

Reference to the inner component of the data.

6.11.3.8 template<class Crtp , class Scalar , class Array , class Tuple > template<class... Dummy, class Type , class , class > const Type & magrathea::AbstractStep< Crtp, Scalar, Array, Tuple >::data (Dummy...) const [inline]

Unified data getter.

Gets the data.

Template Parameters

<i>Dummy</i>	(Dummy types.)
<i>Type</i>	(Data std::tuple<Scalar, Array, Tuple> type.)

Returns

Immutable reference to the data.

6.11.3.9 template<class Crtp , class Scalar , class Array , class Tuple > template<unsigned int Index, unsigned int Subscript, class Type , class , class , class > Crtp & magrathea::AbstractStep< Crtp, Scalar, Array, Tuple >::data (const Type & value) [inline]

Unified data setter.

Unified data inner component setter.

Unified data component setter.

Sets the data.

Parameters

in	value	Data value.
----	-------	-------------

Returns

Self reference.

Sets the specified component of the data.

Template Parameters

<i>Index</i>	Index of the component.
--------------	-------------------------

Parameters

<i>in</i>	<i>value</i>	Component value.
-----------	--------------	------------------

Returns

Self reference.

Sets the specified inner component of the specified component of the data.

Template Parameters

<i>Index</i>	Index of the component.
<i>Subscript</i>	Subscript of the inner component.

Parameters

<i>in</i>	<i>value</i>	Inner component value.
-----------	--------------	------------------------

Returns

Self reference.

6.11.3.10 template<class Crtp , class Scalar , class Array , class Tuple > Crtp & magrathea::AbstractStep< Crtp, Scalar, Array, Tuple >::nullify() [inline]

Nullify.

Resets all data members to their default values.

Returns

Self reference.

6.11.3.11 template<class Crtp , class Scalar , class Array , class Tuple > template<class OtherCrtp , class OtherScalar , class OtherArray , class OtherTuple > bool magrathea::AbstractStep< Crtp, Scalar, Array, Tuple >::operator!= (const AbstractStep< OtherCrtp, OtherScalar, OtherArray, OtherTuple > & rhs) const [inline]

Not equal to.

Compares for difference and returns true if the contents is different.

Template Parameters

<i>OtherCrtp</i>	(Other derived CRTP class.)
<i>OtherScalar</i>	(Other scalar type of id.)
<i>OtherArray</i>	(Other array type of core quantities.)
<i>OtherTuple</i>	(Other tuple type of extra quantities.)

Parameters

<i>in</i>	<i>rhs</i>	Right-hand side.
-----------	------------	------------------

Returns

True if not equal, false if equal.

6.11.3.12 template<class Crtp, class Scalar, class Array, class Tuple> Crtp & magrathea::AbstractStep< Crtp, Scalar, Array, Tuple >::operator= (const AbstractStep< Crtp, Scalar, Array, Tuple > & rhs) [inline]

Copy assignment operator.

Assigns contents from the same type of object.

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Self reference.

6.11.3.13 template<class Crtp , class Scalar , class Array , class Tuple > template<class OtherCrtp , class OtherScalar , class OtherArray , class OtherTuple > Crtp & magrathea::AbstractStep< Crtp, Scalar, Array, Tuple >::operator= (const AbstractStep< OtherCrtp, OtherScalar, OtherArray, OtherTuple > & rhs) [inline]

Conversion assignment operator.

Assigns contents from another type of object.

Template Parameters

<i>OtherCrtp</i>	(Other derived CRTP class.)
<i>OtherScalar</i>	(Other scalar type of id.)
<i>OtherArray</i>	(Other array type of core quantities.)
<i>OtherTuple</i>	(Other tuple type of extra quantities.)

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Self reference.

6.11.3.14 template<class Crtp , class Scalar , class Array , class Tuple > template<class OtherCrtp , class OtherScalar , class OtherArray , class OtherTuple > bool magrathea::AbstractStep< Crtp, Scalar, Array, Tuple >::operator== (const AbstractStep< OtherCrtp, OtherScalar, OtherArray, OtherTuple > & rhs) const [inline]

Equal to.

Compares for equality and returns true if the contents is equal.

Template Parameters

<i>OtherCrtp</i>	(Other derived CRTP class.)
<i>OtherScalar</i>	(Other scalar type of id.)
<i>OtherArray</i>	(Other array type of core quantities.)
<i>OtherTuple</i>	(Other tuple type of extra quantities.)

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

True if equal, false if not equal.

The documentation for this class was generated from the following file:

- /data/home/mbreton/magrathea_pathfinder/src/magrathea/abstractstep.h

6.12 `magrathea::AbstractSubstance< Crtp, Types >` Class Template Reference

Tuple abstraction of geometrical substance.

```
#include <abstractsubstance.h>
```

Public Member Functions

Lifecycle

- **AbstractSubstance ()**
Implicit empty constructor.
 - template<class OtherCrtp , class... OtherTypes>
AbstractSubstance (const **AbstractSubstance**< OtherCrtp, OtherTypes...> &source)
Explicit conversion constructor.
 - template<class... OtherTypes>
AbstractSubstance (const std::tuple< OtherTypes...> &source)
Explicit data constructor.
 - template<class... OtherTypes, class = typename std::enable_if<(sizeof...(OtherTypes) != 0) && (std::is_constructible<typename std::tuple_element<0, typename std::conditional<sizeof...(Types) != 0, std::tuple<Types...>, std::tuple<std::true_type> >::type>-::type, typename std::tuple_element<0, typename std::conditional<sizeof...(OtherTypes) != 0, std::tuple<OtherTypes...>, std::tuple<std::true_type> >::type>::value>>::type>
AbstractSubstance (const OtherTypes &...source)
Explicit components constructor.

Operators

- `Crtp & operator= (const AbstractSubstance< Crtp, Types...> &rhs)`
Copy assignment operator.
 - template<class OtherCrtp , class... OtherTypes>
`Crtp & operator= (const AbstractSubstance< OtherCrtp, OtherTypes...> &rhs)`
Conversion assignment operator.
 - template<class... OtherTypes>
`Crtp & operator= (const std::tuple< OtherTypes...> &rhs)`
Data assignment operator.
 - template<class OtherCrtp , class... OtherTypes>
`bool operator== (const AbstractSubstance< OtherCrtp, OtherTypes...> &rhs) const`
Equal to.
 - template<class OtherCrtp , class... OtherTypes>
`bool operator!= (const AbstractSubstance< OtherCrtp, OtherTypes...> &rhs) const`
Not equal to.

Assignment

- Crtp & `assign ()`
Empty assignment.
 - Crtp & `assign (const AbstractSubstance< Crtp, Types...> &source)`
Copy assignment.

- template<class OtherCrtp , class... OtherTypes>
`Crtp & assign (const AbstractSubstance< OtherCrtp, OtherTypes...> &source)`
Conversion assignment.
- template<class... OtherTypes>
`Crtp & assign (const std::tuple< OtherTypes...> &source)`
Data assignment.
- template<class... OtherTypes, class = typename std::enable_if<(sizeof...(OtherTypes) != 0) && (std::is_constructible<typename std::tuple_element<0, typename std::conditional<sizeof...(Types) != 0, std::tuple<Types...>, std::tuple<std::true_type> >::type>-::type, typename std::tuple_element<0, typename std::conditional<sizeof...(OtherTypes) != 0, std::tuple<OtherTypes...>, std::tuple<std::true_type> >::type>::value>::type>
- `Crtp & assign (const OtherTypes &...source)`
Components assignment.

Management

- `Crtp & nullify ()`
Nullify.
- `Crtp copy () const`
Copy.
- template<class OtherCrtp = Crtp, class = typename std::enable_if<std::is_constructible<OtherCrtp, Crtp>::value>::type>
`OtherCrtp cast () const`
Cast.

Data

- template<class... Dummy, class Type = typename std::conditional<sizeof...(Dummy) == 0, std::tuple<Types...>, void>::type, class = typename std::enable_if<sizeof...(Dummy) == 0>::type, class = typename std::enable_if<std::is_convertible<Type, typename std::conditional<sizeof...(Dummy) == 0, std::tuple<Types...>, void>::type>::value>::type>
`Type & data (Dummy...)`
Unified data access.
- template<class... Dummy, class Type = typename std::conditional<sizeof...(Dummy) == 0, std::tuple<Types...>, void>::type, class = typename std::enable_if<sizeof...(Dummy) == 0>::type, class = typename std::enable_if<std::is_convertible<Type, typename std::conditional<sizeof...(Dummy) == 0, std::tuple<Types...>, void>::type>::value>::type>
`const Type & data (Dummy...) const`
Unified data getter.
- template<class Type , class = typename std::enable_if<std::is_convertible<Type, typename std::conditional<!std::is_void<Type>-::value, std::tuple<Types...>, void>::type>::value>::type>
`Crtp & data (const Type &value)`
Unified data setter.

Protected Member Functions

Protected lifecycle

- `~AbstractSubstance ()`
Protected destructor.

6.12.1 Detailed Description

template<class Crtp, class... Types>class magrathea::AbstractSubstance< Crtp, Types >

Tuple abstraction of geometrical substance.

This class is an abstraction of the substance of geometrical entities. This abstraction provides standardized access to generic types of internal data.

Template Parameters

<i>Crtp</i>	Derived CRTP class.
<i>Types</i>	Variadic list of components types.

6.12.2 Constructor & Destructor Documentation

6.12.2.1 `template<class Crtp , class... Types> magrathea::AbstractSubstance< Crtp, Types >::~AbstractSubstance() [inline], [protected], [default]`

Protected destructor.

Avoids direct instantiation of the class, and only allows it through its derived children.

6.12.2.2 `template<class Crtp , class... Types> magrathea::AbstractSubstance< Crtp, Types >::AbstractSubstance() [inline]`

Implicit empty constructor.

Provides an implicit construction of an object initialized to its default value.

6.12.2.3 `template<class Crtp , class... Types> template<class OtherCrtp , class... OtherTypes> magrathea::AbstractSubstance< Crtp, Types >::AbstractSubstance (const AbstractSubstance< OtherCrtp, OtherTypes...> & source) [inline], [explicit]`

Explicit conversion constructor.

Provides an explicit construction from another type of object.

Template Parameters

<i>OtherCrtp</i>	(Other derived CRTP class.)
<i>OtherTypes</i>	(Other variadic list of components types.)

Parameters

<i>in</i>	<i>source</i>	Source of the copy.
-----------	---------------	---------------------

6.12.2.4 `template<class Crtp , class... Types> template<class... OtherTypes> magrathea::AbstractSubstance< Crtp, Types >::AbstractSubstance (const std::tuple< OtherTypes...> & source) [inline], [explicit]`

Explicit data constructor.

Provides an explicit construction from data.

Template Parameters

<i>OtherTypes</i>	(Other variadic list of object property types.)
-------------------	---

Parameters

<i>in</i>	<i>source</i>	Source of the copy.
-----------	---------------	---------------------

6.12.2.5 template<class Crtp , class... Types> template<class... OtherTypes, class > **magrathea::AbstractSubstance< Crtp, Types >::AbstractSubstance** (const OtherTypes &... *source*) [inline], [explicit]

Explicit components constructor.

Provides an explicit construction from components.

Template Parameters

<i>OtherTypes</i>	(Other variadic list of object property types.)
-------------------	---

Parameters

in	<i>source</i>	Source of the copy.
----	---------------	---------------------

6.12.3 Member Function Documentation

6.12.3.1 template<class Crtp , class... Types> Crtp & **magrathea::AbstractSubstance< Crtp, Types >::assign** () [inline]

Empty assignment.

Assigns contents from an object initialized to its default value.

Returns

Self reference.

6.12.3.2 template<class Crtp, class... Types> Crtp & **magrathea::AbstractSubstance< Crtp, Types >::assign** (const AbstractSubstance< Crtp, Types...> & *source*) [inline]

Copy assignment.

Assigns contents from the same type of object.

Parameters

in	<i>source</i>	Source of the copy.
----	---------------	---------------------

Returns

Self reference.

6.12.3.3 template<class Crtp , class... Types> template<class OtherCrtp , class... OtherTypes> Crtp & **magrathea::AbstractSubstance< Crtp, Types >::assign** (const AbstractSubstance< OtherCrtp, OtherTypes...> & *source*) [inline]

Conversion assignment.

Assigns contents from another type of object.

Template Parameters

<i>OtherCrtp</i>	(Other derived CRTP class.)
<i>OtherTypes</i>	(Other variadic list of components types.)

Parameters

in	source	Source of the copy.
----	--------	---------------------

Returns

Self reference.

6.12.3.4 template<class Crtp , class... Types> template<class... OtherTypes> Crtp & magrathea::AbstractSubstance<Crtp, Types >::assign (const std::tuple< OtherTypes...> & source) [inline]

Data assignment.

Assigns contents from data.

Template Parameters

<i>OtherTypes</i>	(Other variadic list of object property types.)
-------------------	---

Parameters

in	source	Source of the copy.
----	--------	---------------------

Returns

Self reference.

6.12.3.5 template<class Crtp , class... Types> template<class... OtherTypes, class > Crtp & magrathea::AbstractSubstance< Crtp, Types >::assign (const OtherTypes &... source) [inline]

Components assignment.

Assigns contents from components.

Template Parameters

<i>OtherTypes</i>	(Other variadic list of object property types.)
-------------------	---

Parameters

in	source	Source of the copy.
----	--------	---------------------

Returns

Self reference.

6.12.3.6 template<class Crtp , class... Types> template<class OtherCrtp , class > OtherCrtp magrathea::AbstractSubstance< Crtp, Types >::cast () const [inline]

Cast.

Casts contents to another object type.

Template Parameters

<i>OtherCrtp</i>	Other derived CRTP class.
------------------	---------------------------

Returns

Casted copy.

6.12.3.7 template<class Crtp , class... Types> Crtp magrathea::AbstractSubstance< Crtp, Types >::copy () const [inline]

Copy.

Generates a copy of the object.

Returns

Copy.

6.12.3.8 template<class Crtp , class... Types> template<class... Dummy, class Type , class , class > Type & magrathea::AbstractSubstance< Crtp, Types >::data (Dummy...) [inline]

Unified data access.

Unified data inner component access.

Unified data component access.

Provides a direct access to the data.

Template Parameters

<i>Dummy</i>	(Dummy types.)
<i>Type</i>	(Data std::tuple<Types...> type.)

Returns

Reference to the data.

Provides a direct access to the specified component of the data.

Template Parameters

<i>Index</i>	Index of the component.
<i>Dummy</i>	(Dummy types.)
<i>Type</i>	(Component type.)

Returns

Reference to the component of the data.

Provides a direct access to the specified inner component of the specified component of the data.

Template Parameters

<i>Index</i>	Index of the component.
<i>Subscript</i>	Subscript of the inner component.
<i>Dummy</i>	(Dummy types.)
<i>Type</i>	(Inner component type.)

Returns

Reference to the inner component of the data.

6.12.3.9 template<class Crtp , class... Types> template<class... Dummy, class Type , class , class > const Type & magrathea::AbstractSubstance< Crtp, Types >::data (Dummy...) const [inline]

Unified data getter.

Gets the data.

Template Parameters

<i>Dummy</i>	(Dummy types.)
<i>Type</i>	(Data std::tuple<Types...> type.)

Returns

Immutable reference to the data.

6.12.3.10 template<class Crtp , class... Types> template<unsigned int Index, unsigned int Subscript, class Type , class , class > Crtp & magrathea::AbstractSubstance< Crtp, Types >::data (const Type & value) [inline]

Unified data setter.

Unified data inner component setter.

Unified data component setter.

Sets the data.

Parameters

<i>in</i>	<i>value</i>	Data value.
-----------	--------------	-------------

Returns

Self reference.

Sets the specified component of the data.

Template Parameters

<i>Index</i>	Index of the component.
--------------	-------------------------

Parameters

<i>in</i>	<i>value</i>	Component value.
-----------	--------------	------------------

Returns

Self reference.

Sets the specified inner component of the specified component of the data.

Template Parameters

<i>Index</i>	Index of the component.
<i>Subscript</i>	Subscript of the inner component.

Parameters

in	<i>value</i>	Inner component value.
----	--------------	------------------------

Returns

Self reference.

6.12.3.11 template<class Crtp , class... Types> Crtp & magrathea::AbstractSubstance< Crtp, Types >::nullify () [inline]

Nullify.

Resets all data members to their default values.

Returns

Self reference.

6.12.3.12 template<class Crtp , class... Types> template<class OtherCrtp , class... OtherTypes> bool magrathea::AbstractSubstance< Crtp, Types >::operator!= (const AbstractSubstance< OtherCrtp, OtherTypes...> & rhs) const [inline]

Not equal to.

Compares for difference and returns true if the contents is different.

Template Parameters

<i>OtherCrtp</i>	(Other derived CRTP class.)
<i>OtherTypes</i>	(Other variadic list of components types.)

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

True if not equal, false if equal.

6.12.3.13 template<class Crtp, class... Types> Crtp & magrathea::AbstractSubstance< Crtp, Types >::operator= (const AbstractSubstance< Crtp, Types...> & rhs) [inline]

Copy assignment operator.

Assigns contents from the same type of object.

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Self reference.

6.12.3.14 template<class Crtp , class... Types> template<class OtherCrtp , class... OtherTypes> Crtp &
**magrathea::AbstractSubstance< Crtp, Types >::operator= (const AbstractSubstance< OtherCrtp,
 OtherTypes...> & rhs) [inline]**

Conversion assignment operator.

Assigns contents from another type of object.

Template Parameters

<i>OtherCrtp</i>	(Other derived CRTP class.)
<i>OtherTypes</i>	(Other variadic list of components types.)

Parameters

<i>in</i>	<i>rhs</i>	Right-hand side.
-----------	------------	------------------

Returns

Self reference.

6.12.3.15 template<class Crtp , class... Types> template<class... OtherTypes> Crtp & **magrathea::-**
AbstractSubstance< Crtp, Types >::operator= (const std::tuple< OtherTypes...> & rhs)
 [inline]

Data assignment operator.

Assigns contents from data.

Template Parameters

<i>OtherTypes</i>	(Other variadic list of object property types.)
-------------------	---

Parameters

<i>in</i>	<i>rhs</i>	Right-hand side.
-----------	------------	------------------

Returns

Self reference.

6.12.3.16 template<class Crtp , class... Types> template<class OtherCrtp , class... OtherTypes> bool
**magrathea::AbstractSubstance< Crtp, Types >::operator== (const AbstractSubstance< OtherCrtp,
 OtherTypes...> & rhs) const [inline]**

Equal to.

Compares for equality and returns true if the contents is equal.

Template Parameters

<i>OtherCrtp</i>	(Other derived CRTP class.)
<i>OtherTypes</i>	(Other variadic list of components types.)

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

True if equal, false if not equal.

The documentation for this class was generated from the following file:

- /data/home/mbreton/magrathea_pathfinder/src/magrathea/abstractsubstance.h

6.13 Catalogues Class Reference

```
#include <catalogues.h>
```

Static Public Member Functions

- template<class Parameters , class Map >
static void ReadParamFile (Parameters ¶meters, Map ¶meter)
Read parameter file.
- template<typename Integer , class Parameter >
static void ReadParticlesHDF5 (const Integer rank, const Parameter ¶meters, std::vector< std::array< double, 8 > > &caractVect_source)
Read Particles from HDF5 files.
- template<typename Integer , class Parameter >
static void ReadParticlesASCII (const Integer rank, const Parameter ¶meters, std::vector< std::array< double, 8 > > &caractVect_source)
Read Particles from ASCII files.
- template<int Order = ORDER, bool RK4 = true, bool Verbose = false, class Point , class Cosmology , class Octree , class Type , class Parameter >
static std::array< std::array< double, 2 >, 2 > newtonMethod2d (const Point &vobs, const Point &observer, const Point &trueTarget, const std::array< double, 2 > &target, const Point &velocity, std::array< std::array< double, 2 >, 2 > &jacobian, const Parameter ¶meters, const Cosmology &cosmology, const Octree &octree, const Type length, const Type h, std::vector< double > &redshifts, double &interpRef, const unsigned int iteration)
Iterate once using Newton Method.
- template<int Order = ORDER, bool RK4 = true, bool Verbose = false, class Point , class Cosmology , class Octree , class Type , class Parameter >
static std::array< std::array< double, 2 >, 2 > iterateNewtonMethod (const Point &vobs, const Point &observer, const Type phi, const Type theta, const Point &target, const Point &velocity, std::array< std::array< double, 2 >, 2 > &jacobian, const Parameter ¶meters, const Cosmology &cosmology, const Octree &octree, const Type length, const Type h, std::vector< double > &redshifts, double &interpRef)
Iterate Newton Method.
- template<class Point , class Cosmology , class Octree , class Type , class Parameter >
static void relCat (const Point &vobs, const std::array< std::array< double, 3 >, 3 > &rotm1, std::string &nomOutput, const Point &observer, const std::vector< std::array< double, 8 > > &targets_position, const std::vector< std::array< double, 18 > > &previous_catalogue, const Parameter ¶meters, const Cosmology &cosmology, const Octree &octree, const Type length, const Type h)
Give new position of points.
- template<class Point , class Cosmology , class Octree , class Type , class Parameter >
static void relCat_with_previous_cat (const Point &vobs, std::string &nomOutput, const Point &observer, std::vector< std::array< double, 18 > > &previous_catalogue, const Parameter ¶meters, const Cosmology &cosmology, const Octree &octree, const Type length, const Type h)
Give new position of points.

6.13.1 Member Function Documentation

6.13.1.1 template<int Order, bool RK4, bool Verbose, class Point , class Cosmology , class Octree , class Type , class Parameter > std::array< std::array< double, 2 >, 2 > Catalogues::iterateNewtonMethod (const Point & vobs, const Point & observer, const Type phi, const Type theta, const Point & target, const Point & velocity, std::array< std::array< double, 2 >, 2 > & jacobian, const Parameter & parameters, const Cosmology & cosmology, const Octree & octree, const Type length, const Type h, std::vector< double > & redshifts, double & interpRef) [static]

Iterate Newton Method.

Iterate Newton Method to a certain precision

Template Parameters

<i>Order</i>	Octree interpolation order : 0 for NGP, 1 for CIC, 2 for TSC or -1 for a homogeneous universe.
<i>RK4</i>	Runge-kutta of fourth order or euler.
<i>Verbose</i>	Verbose mode for debug purposes.
<i>Point</i>	point type.
<i>Cosmology</i>	Cosmology evolution type.
<i>Octree</i>	Octree type.
<i>Type</i>	Scalar type.
<i>Parameter</i>	Parameter type.

Parameters

in	<i>vobs</i>	Observer peculiar velocity, in SI
in	<i>observer</i>	Observer position.
in	<i>phi</i>	Source comoving angular coordinate phi.
in	<i>theta</i>	Source comoving angular coordinate theta.
in	<i>target</i>	Comoving position of the source
in	<i>velocity</i>	Velocity of the target.
in,out	<i>jacobian</i>	Jacobian matrix
in	<i>parameters</i>	Parameter structure
in	<i>cosmology</i>	Cosmology evolution.
in	<i>octree</i>	Octree.
in	<i>length</i>	Spatial length in SI units.
in	<i>h</i>	Dimensionless Hubble parameter
in,out	<i>redshifts</i>	Vector containing the redshift decomposition for a given source.
in	<i>interpRef</i>	value for interpolation of the bundle.

Returns

2x2 array with NEW initial angles given after several iterations of newton method and errors in angles at same radius for the NEW initial angles.

6.13.1.2 template<int Order, bool RK4, bool Verbose, class Point , class Cosmology , class Octree , class Type , class Parameter > std::array< std::array< double, 2 >, 2 > Catalogues::newtonMethod2d (const Point & vobs, const Point & observer, const Point & trueTarget, const std::array< double, 2 > & target, const Point & velocity, std::array< std::array< double, 2 >, 2 > & jacobian, const Parameter & parameters, const Cosmology & cosmology, const Octree & octree, const Type length, const Type h, std::vector< double > & redshifts, double & interpRef, const unsigned int iteration) [static]

Iterate once using Newton Method.

Fit the better initial conditions for the photon that intersects the source.

Template Parameters

<i>Order</i>	Octree interpolation order : 0 for NGP, 1 for CIC, 2 for TSC or -1 for an homogeneous universe.
<i>RK4</i>	Runge-kutta of fourth order or euler.
<i>Verbose</i>	Verbose mode for debug purposes.
<i>Point</i>	point type.
<i>Cosmology</i>	Cosmology evolution type.
<i>Octree</i>	Octree type.
<i>Type</i>	Scalar type.
<i>Parameter</i>	Parameter type

Parameters

in	<i>vobs</i>	Observer peculiar velocity, in SI
in	<i>observer</i>	Observer cartesian position.
in	<i>trueTarget</i>	True cartesian target position.
in	<i>target</i>	Observed angular target position.
in	<i>velocity</i>	Target velocity.
in,out	<i>jacobian</i>	Jacobian matrix
in	<i>parameters</i>	Parameters structure
in	<i>cosmology</i>	Cosmology evolution.
in	<i>octree</i>	Octree.
in	<i>length</i>	Spatial length in SI units.
in	<i>h</i>	Dimensionless Hubble parameter
in,out	<i>redshifts</i>	Vector containing the redshift decomposition for a given source.
in,out	<i>interpRef</i>	value for interpolation of the bundle.
in	<i>iteration</i>	Number of iterations for the root-finder

Returns

2x2 array with NEW initial angles given by newton method and angle difference at the source between source and photon

6.13.1.3 template<class Parameters , class Map > void Catalogues::ReadParamFile (Parameters & parameters, Map & parameter) [static]

Read parameter file.

Read and put in a structure the parameters.

Template Parameters

<i>Parameters</i>	structure type
<i>Map</i>	map type

Parameters

in,out	<i>parameters</i>	Structure containing the parameters.
in	<i>parameter</i>	Contains parameters to be rewritten

Returns

Filled parameters structure.

6.13.1.4 template<typename Integer , class Parameter > void Catalogues::ReadParticlesASCII (const Integer *rank*, const Parameter & *parameters*, std::vector< std::array< double, 8 > > & *caractVect_source*) [static]

Read Particles from ASCII files.

Read particles from ASCII files and put them in vectors

Template Parameters

<i>Integer</i>	Integer type
<i>Parameter</i>	Parameter type

Parameters

in	<i>rank</i>	Rank
in	<i>parameters</i>	Parameters structure
in	<i>caractVect_source</i>	Source characteristics

6.13.1.5 template<typename Integer , class Parameter > void Catalogues::ReadParticlesHDF5 (const Integer *rank*, const Parameter & *parameters*, std::vector< std::array< double, 8 > > & *caractVect_source*) [static]

Read Particles from HDF5 files.

Read particles from HDF5 files and put them in vectors

Template Parameters

<i>Integer</i>	Integer type
<i>Parameter</i>	Parameter type

Parameters

in	<i>rank</i>	Rank
in	<i>parameters</i>	Parameters structure
in	<i>caractVect_source</i>	Source characteristics

6.13.1.6 template<class Point , class Cosmology , class Octree , class Type , class Parameter > void Catalogues::relCat (const Point & *vobs*, const std::array< std::array< double, 3 >, 3 > & *rotm1*, std::string & *filename*, const Point & *observer*, const std::vector< std::array< double, 8 > > & *targets_position*, const std::vector< std::array< double, 18 > > & *previous_catalogue*, const Parameter & *parameters*, const Cosmology & *cosmology*, const Octree & *octree*, const Type *length*, const Type *h*) [static]

Give new position of points.

Give lensed position of points given points.

Template Parameters

<i>Point</i>	point type.
<i>Cosmology</i>	Cosmology evolution type.
<i>Octree</i>	Octree type.
<i>Type</i>	Scalar type.
<i>Parameter</i>	Parameter type.

Parameters

in	<i>vobs</i>	Observer peculiar velocity, in SI
in	<i>rotm1</i>	rotation matrix
in	<i>filename</i>	output file name.
in	<i>observer</i>	Observer position.
in	<i>targets_position</i>	vector of target positions, velocities and radius.
in	<i>previous_catalogue</i>	Previously computed catalogue, used to re-run rejected sources.
in	<i>parameters</i>	Parameter structure
in	<i>cosmology</i>	Cosmology evolution.
in	<i>octree</i>	Octree.
in	<i>length</i>	Spatial length in SI units.
in	<i>h</i>	Dimensionless Hubble parameter

```
6.13.1.7 template<class Point , class Cosmology , class Octree , class Type , class Parameter > void
Catalogues::relCat_with_previous_cat ( const Point & vobs, std::string & filename, const Point & observer,
std::vector< std::array< double, 18 > > & previous_catalogue, const Parameter & parameters, const Cosmology &
cosmology, const Octree & octree, const Type length, const Type h ) [static]
```

Give new position of points.

Give lensed position of points given points.

Template Parameters

<i>Point</i>	point type.
<i>Cosmology</i>	Cosmology evolution type.
<i>Octree</i>	Octree type.
<i>Type</i>	Scalar type.
<i>Parameter</i>	Parameter type.

Parameters

in	<i>vobs</i>	Observer peculiar velocity, in SI
in	<i>filename</i>	output file name.
in	<i>observer</i>	Observer position.
in	<i>previous_catalogue</i>	Previously computed catalogue, used to re-run rejected sources.
in	<i>parameters</i>	Parameter structure
in	<i>cosmology</i>	Cosmology evolution.
in	<i>octree</i>	Octree.
in	<i>length</i>	Spatial length in SI units.
in	<i>h</i>	Dimensionless Hubble parameter

The documentation for this class was generated from the following file:

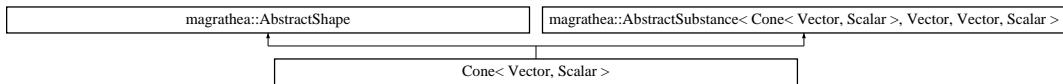
- /data/home/mbreton/magrathea_pathfinder/src/[catalogues.h](#)

6.14 Cone< Vector, Scalar > Exception Template Reference

Three-dimensional cone.

```
#include <cone.h>
```

Inheritance diagram for Cone< Vector, Scalar >:



Public Member Functions

Lifecycle

- template<class... Misc>
Cone (Misc &&...misc)
Explicit generic constructor.

Data

- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<magrathea::AbstractSubstance<Cone<Vector, Scalar>, Vector, Vector, Vector, Scalar>>().template data<0, Values...>(std::declval<Misc>(...))), class = typename std::enable_if<!std::is_void<Template>::value>::type>
Template vertex (Misc &&...misc)
Access to the vertex data.
- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<const magrathea::AbstractSubstance<Cone<Vector, Scalar>, Vector, Vector, Vector, Scalar>>().template data<0, Values...>(std::declval<Misc>(...))), class = typename std::enable_if<!std::is_void<Template>::value>::type>
Template vertex (Misc &&...misc) const
Immutable access to the vertex data.
- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<magrathea::AbstractSubstance<Cone<Vector, Scalar>, Vector, Vector, Vector, Scalar>>().template data<1, Values...>(std::declval<Misc>(...))), class = typename std::enable_if<!std::is_void<Template>::value>::type>
Template base (Misc &&...misc)
Access to the base data.
- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<const magrathea::AbstractSubstance<Cone<Vector, Scalar>, Vector, Vector, Vector, Scalar>>().template data<1, Values...>(std::declval<Misc>(...))), class = typename std::enable_if<!std::is_void<Template>::value>::type>
Template base (Misc &&...misc) const
Immutable access to the base data.
- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<magrathea::AbstractSubstance<Cone<Vector, Scalar>, Vector, Vector, Vector, Scalar>>().template data<2, Values...>(std::declval<Misc>(...))), class = typename std::enable_if<!std::is_void<Template>::value>::type>
Template angle (Misc &&...misc)
Access to the angle data.
- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<const magrathea::AbstractSubstance<Cone<Vector, Scalar>, Vector, Vector, Vector, Scalar>>().template data<2, Values...>(std::declval<Misc>(...))), class = typename std::enable_if<!std::is_void<Template>::value>::type>
Template angle (Misc &&...misc) const
Immutable access to the angle data.

Position

- Scalar **direction** (const unsigned int idim) const
Direction coordinate.
- Vector **direction** () const
Direction vector.

Measures

- Scalar **length** () const
Length.
- Scalar **radius** () const

- Scalar **diameter** () const
Diameter.
- Scalar **circle** () const
Circle.
- Scalar **surface** () const
Surface.
- Scalar **volume** () const
Volume.

Collision

- template<class OtherVector , class = typename std::enable_if<std::is_convertible<typename std::remove_cv<typename std::remove_reference<decltype(std::declval<OtherVector>()[0])>::type>::type, Scalar>::value>::type>
bool **inside** (const OtherVector &**point**) const
Point inside.
- template<class OtherVector , class = typename std::enable_if<std::is_convertible<typename std::remove_cv<typename std::remove_reference<decltype(std::declval<OtherVector>()[0])>::type>::type, Scalar>::value>::type>
bool **outside** (const OtherVector &**point**) const
Point outside.

Static Public Member Functions

Test

- static int **example** ()
Example function.

Public Attributes

- using **operator** = typedef

Additional Inherited Members

6.14.1 Detailed Description

```
template<class Vector = std::array<double, 3>, typename Scalar = typename std::remove_cv<typename std::remove_reference<decltype(std::declval<Vector>()[0])>::type>::type>exception Cone< Vector, Scalar >
```

Three-dimensional cone.

Implementation of a basic cone in three dimensions.

Template Parameters

Vector	Position vector type.
Scalar	Scalar data type.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 template<class Vector , typename Scalar > template<class... Misc> Cone< Vector, Scalar >::Cone (*Misc &&... misc*) [inline], [explicit]

Explicit generic constructor.

Provides a generic interface to all constructors of the base class.

Template Parameters

<i>Misc</i>	(Miscellaneous types.)
-------------	------------------------

Parameters

<i>in</i>	<i>misc</i>	Miscellaneous arguments.
-----------	-------------	--------------------------

6.14.3 Member Function Documentation

6.14.3.1 template<class Vector , typename Scalar > template<unsigned int... Values, class... Misc, class Template , class >
Template Cone< Vector, Scalar >::angle (Misc &&... misc) [inline]

Access to the angle data.

Provides an access to the angle data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

<i>in</i>	<i>misc</i>	Miscellaneous arguments.
-----------	-------------	--------------------------

Returns

Forwarded result.

6.14.3.2 template<class Vector , typename Scalar > template<unsigned int... Values, class... Misc, class Template , class >
Template Cone< Vector, Scalar >::angle (Misc &&... misc) const [inline]

Immutable access to the angle data.

Provides an immutable access to the angle data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

<i>in</i>	<i>misc</i>	Miscellaneous arguments.
-----------	-------------	--------------------------

Returns

Forwarded result.

6.14.3.3 template<class Vector , typename Scalar > template<unsigned int... Values, class... Misc, class Template , class > Template Cone< Vector, Scalar >::base (Misc &&... misc) [inline]

Access to the base data.

Provides an access to the base data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.14.3.4 template<class Vector , typename Scalar > template<unsigned int... Values, class... Misc, class Template , class > Template Cone< Vector, Scalar >::base (Misc &&... misc) const [inline]

Immutable access to the base data.

Provides an immutable access to the base data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.14.3.5 template<class Vector , typename Scalar > Scalar Cone< Vector, Scalar >::circle () const [inline]

Circle.

Computes the base area of the cone.

Returns

B.

6.14.3.6 template<class Vector , typename Scalar > Scalar Cone< Vector, Scalar >::diameter () const [inline]

Diameter.

Computes the base diameter of the cone.

Returns

d.

6.14.3.7 template<class Vector , typename Scalar > Scalar Cone< Vector, Scalar >::direction (const unsigned int *idim*)
const [inline]

Direction coordinate.

Computes the specified coordinate of the vector going from the vertex to the base center.

Parameters

in	<i>idim</i>	Index of the dimension.
----	-------------	-------------------------

Returns

The coordinate of $b_i - v_i$.

6.14.3.8 template<class Vector , typename Scalar > Vector Cone< Vector, Scalar >::direction () const [inline]

Direction vector.

Computes the vector going from the vertex to the base center.

Returns

The vector $\vec{b} - \vec{v}$.

6.14.3.9 template<class Vector , typename Scalar > int Cone< Vector, Scalar >::example () [static]

Example function.

Tests and demonstrates the use of [Cone](#).

Returns

0 if no error.

6.14.3.10 template<class Vector , typename Scalar > template<class OtherVector , class > bool Cone< Vector, Scalar >::inside (const OtherVector & *point*) const [inline]

Point inside.

Checks whether a point is inside the cone.

Template Parameters

OtherVector	Other position vector type.
-------------	-----------------------------

Parameters

in	<i>point</i>	Position of the point.
----	--------------	------------------------

Returns

True if the point is inside the cone, false otherwise.

6.14.3.11 template<class Vector , typename Scalar > Scalar Cone< Vector, Scalar >::length () const [inline]

Length.

Computes the height length of the cone.

Returns

h .

6.14.3.12 template<class Vector , typename Scalar > template<class OtherVector , class > bool Cone< Vector, Scalar >::outside (const OtherVector & point) const [inline]

Point outside.

Checks whether a point is outside the cone.

Template Parameters

<i>OtherVector</i>	Other position vector type.
--------------------	-----------------------------

Parameters

in	<i>point</i>	Position of the point.
----	--------------	------------------------

Returns

True if the point is outside the cone, false otherwise.

6.14.3.13 template<class Vector , typename Scalar > Scalar Cone< Vector, Scalar >::radius () const [inline]

Radius.

Computes the base radius of the cone.

Returns

r .

6.14.3.14 template<class Vector , typename Scalar > Scalar Cone< Vector, Scalar >::surface () const [inline]

Surface.

Computes the outer surface of the cone.

Returns

$\pi \times r^2 + \pi \times r \times h$.

6.14.3.15 template<class Vector , typename Scalar > template<unsigned int... Values, class... Misc, class Template , class > Template Cone< Vector, Scalar >::vertex (*Misc &&... misc*) [inline]

Access to the vertex data.

Provides an access to the vertex data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.14.3.16 template<class Vector , typename Scalar > template<unsigned int... Values, class... Misc, class Template , class > Template Cone< Vector, Scalar >::vertex (*Misc &&... misc*) const [inline]

Immutable access to the vertex data.

Provides an immutable access to the vertex data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.14.3.17 template<class Vector , typename Scalar > Scalar Cone< Vector, Scalar >::volume () const [inline]

Volume.

Computes the volume of the cone.

Returns

$$\frac{\pi \times r^2 \times h}{3}.$$

6.14.4 Member Data Documentation

```
6.14.4.1 template<class Vector = std::array<double, 3>, typename Scalar = typename std::remove_cv<typename std::remove_reference<decltype(std::declval<Vector>()[0])>::type>::type> using Cone< Vector, Scalar >::operator =
```

The documentation for this exception was generated from the following file:

- /data/home/mbreton/magrathea_pathfinder/src/cone.h

6.15 magrathea::Constant< Type, Size > Exception Template Reference

Numerical constant with `constexpr` constructor.

```
#include <constant.h>
```

Public Member Functions

Lifecycle

- `constexpr Constant (const Type source)`
Implicit constexpr value constructor.
- `template<typename... Types, class = typename std::enable_if<sizeof...(Types) != 1>::type> constexpr Constant (const Types...source)`
Implicit constexpr values constructor.
- `template<typename OtherType, class = typename std::enable_if<std::is_convertible<OtherType, Type>::value>::type> constexpr Constant (const OtherType source)`
Implicit constexpr value conversion constructor.
- `template<typename OtherType, class = typename std::enable_if<std::is_convertible<OtherType, Type>::value>::type> constexpr Constant (const Constant< OtherType, Size > source)`
Implicit constexpr constant conversion constructor.
- `template<typename OtherType, class = typename std::enable_if<std::is_convertible<OtherType, Type>::value>::type> constexpr Constant (const std::array< OtherType, Size > source)`
Implicit constexpr array conversion constructor.
- `constexpr Constant (const std::array< Type, Size > source)`
Implicit constexpr array constructor.

Operators

- `template<typename... Types, class = typename std::enable_if<sizeof...(Types) != Size>::type> constexpr const Constant< Type, Size > operator() (const Types ...values) const`
Incomplete setter operator.
- `template<typename... Types, class = typename std::enable_if<sizeof...(Types) == Size>::type> constexpr const Constant< Type, Size > operator() (const Types...values) const`
Setter operator.
- `constexpr Type operator() () const`
Default getter operator.
- `constexpr Type operator[](const unsigned int i) const`
Subscript operator.
- `constexpr operator Type () const`
Value cast operator.

Management

- `constexpr unsigned int size () const`
Size.

- template<unsigned int OtherSize>
constexpr const **Constant**< Type,
OtherSize > **resize** () const
Resize.
- constexpr const **Constant**< Type,
Size > **nullify** () const
Nullify.
- template<typename... Types, class = typename std::enable_if<sizeof...(Types) != Size>::type>
constexpr const **Constant**< Type,
Size > **set** (const Types &...values) const
Incomplete value setter.
- template<typename... Types, class = typename std::enable_if<sizeof...(Types) == Size>::type>
constexpr const **Constant**< Type,
Size > **set** (const Types...values) const
Value setter.
- template<typename OtherType = Type>
constexpr OtherType **get** (const unsigned int i=0) const
Value getter.
- template<typename OtherType = Type>
constexpr OtherType **value** (const unsigned int i=0) const
Access a value.
- template<typename OtherType = Type>
constexpr const std::array
< OtherType, Size > **data** () const
Access data.
- template<typename OtherType >
constexpr const **Constant**
< OtherType, Size > **cast** () const
Cast.
- constexpr const **Constant**< Type,
Size > **copy** () const
Copy.

Unary operations

- template<typename... Types, class = typename std::enable_if<sizeof...(Types) != Size>::type>
constexpr const **Constant**< Type,
Size > **inv** (const Types &...values) const
Start calculation of the inverse.
- template<typename... Types, class = typename std::enable_if<sizeof...(Types) == Size>::type>
constexpr const **Constant**< Type,
Size > **inv** (const Types...values) const
Stop calculation of the inverse.
- template<typename... Types, class = typename std::enable_if<sizeof...(Types) != Size>::type>
constexpr const **Constant**< Type,
Size > **opp** (const Types &...values) const
Start calculation of the opposite.
- template<typename... Types, class = typename std::enable_if<sizeof...(Types) == Size>::type>
constexpr const **Constant**< Type,
Size > **opp** (const Types...values) const
Stop calculation of opposite.
- template<typename... Types, class = typename std::enable_if<sizeof...(Types) != Size>::type>
constexpr const **Constant**< Type,
Size > **abs** (const Types &...values) const
Start calculation of the absolute value.
- template<typename... Types, class = typename std::enable_if<sizeof...(Types) == Size>::type>
constexpr const **Constant**< Type,
Size > **abs** (const Types...values) const
Stop calculation of the absolute value.

- template<typename... Types, class = typename std::enable_if<sizeof...(Types) != Size>::type>
constexpr const Constant< Type,
Size > sgn (const Types &...values) const
Start calculation of the the signum function.
- template<typename... Types, class = typename std::enable_if<sizeof...(Types) == Size>::type>
constexpr const Constant< Type,
Size > sgn (const Types...values) const
Stop calculation of the the signum function.
- template<typename... Types, class = typename std::enable_if<sizeof...(Types) != Size>::type>
constexpr const Constant< Type,
Size > sq (const Types &...values) const
Start calculation of the square.
- template<typename... Types, class = typename std::enable_if<sizeof...(Types) == Size>::type>
constexpr const Constant< Type,
Size > sq (const Types...values) const
Stop calculation of the square.
- template<typename... Types, class = typename std::enable_if<sizeof...(Types) != Size>::type>
constexpr const Constant< Type,
Size > cb (const Types &...values) const
Start calculation of the cube.
- template<typename... Types, class = typename std::enable_if<sizeof...(Types) == Size>::type>
constexpr const Constant< Type,
Size > cb (const Types...values) const
Stop calculation of the cube.
- template<typename... Types, class = typename std::enable_if<sizeof...(Types) != Size>::type>
constexpr const Constant< Type,
Size > isqrt (const Types &...values) const
Start calculation of the integer square root.
- template<typename... Types, class = typename std::enable_if<sizeof...(Types) == Size>::type>
constexpr const Constant< Type,
Size > isqrt (const Types...values) const
Stop calculation of the integer square root.
- template<typename... Types, class = typename std::enable_if<sizeof...(Types) != Size>::type>
constexpr const Constant< Type,
Size > icbrt (const Types &...values) const
Start calculation of the integer cube root.
- template<typename... Types, class = typename std::enable_if<sizeof...(Types) == Size>::type>
constexpr const Constant< Type,
Size > icbrt (const Types...values) const
Stop calculation of the integer cube root.
- template<typename... Types, class = typename std::enable_if<sizeof...(Types) != Size>::type>
constexpr const Constant< Type,
Size > ilog2 (const Types &...values) const
Start calculation of the integer base 2 logarithm.
- template<typename... Types, class = typename std::enable_if<sizeof...(Types) == Size>::type>
constexpr const Constant< Type,
Size > ilog2 (const Types...values) const
Stop calculation of the integer base 2 logarithm.
- template<typename... Types, class = typename std::enable_if<sizeof...(Types) != Size>::type>
constexpr const Constant< Type,
Size > ilog10 (const Types &...values) const
Start calculation of the integer base 10 logarithm.
- template<typename... Types, class = typename std::enable_if<sizeof...(Types) == Size>::type>
constexpr const Constant< Type,
Size > ilog10 (const Types...values) const
Stop calculation of the integer base 10 logarithm.
- template<class Ratio = std::ratio<1>, typename... Types, class = typename std::enable_if<sizeof...(Types) != Size>::type>
constexpr const Constant< Type,
Size > si (const Types &...values) const

Start multiplication by a SI prefix.

- template<class Ratio = std::ratio<1>, typename... Types, class = typename std::enable_if<sizeof...(Types) == Size>::type> constexpr const Constant< Type, Size > **si** (const Types...values) const

Stop multiplication by a SI prefix.

Arithmetic operations

- template<typename... Types, class = typename std::enable_if<sizeof...(Types) != Size>::type> constexpr const Constant< Type, Size > **add** (const Type rhs, const Types &...values) const

Start addition.
- template<typename... Types, class = typename std::enable_if<sizeof...(Types) == Size>::type> constexpr const Constant< Type, Size > **add** (const Type, const Types...values) const

Stop addition.
- template<typename... Types, class = typename std::enable_if<sizeof...(Types) != Size>::type> constexpr const Constant< Type, Size > **sub** (const Type rhs, const Types &...values) const

Start subtraction.
- template<typename... Types, class = typename std::enable_if<sizeof...(Types) == Size>::type> constexpr const Constant< Type, Size > **sub** (const Type, const Types...values) const

Stop subtraction.
- template<typename... Types, class = typename std::enable_if<sizeof...(Types) != Size>::type> constexpr const Constant< Type, Size > **mul** (const Type rhs, const Types &...values) const

Start multiplication.
- template<typename... Types, class = typename std::enable_if<sizeof...(Types) == Size>::type> constexpr const Constant< Type, Size > **mul** (const Type, const Types...values) const

Stop multiplication.
- template<typename... Types, class = typename std::enable_if<sizeof...(Types) != Size>::type> constexpr const Constant< Type, Size > **div** (const Type rhs, const Types &...values) const

Start division.
- template<typename... Types, class = typename std::enable_if<sizeof...(Types) == Size>::type> constexpr const Constant< Type, Size > **div** (const Type, const Types...values) const

Stop division.
- template<typename... Types, class = typename std::enable_if<sizeof...(Types) != Size>::type> constexpr const Constant< Type, Size > **mod** (const Type rhs, const Types &...values) const

Start modulo calculation.
- template<typename... Types, class = typename std::enable_if<sizeof...(Types) == Size>::type> constexpr const Constant< Type, Size > **mod** (const Type, const Types...values) const

Stop modulo calculation.

Mathematical functions

- template<typename... Types, class = typename std::enable_if<sizeof...(Types) != Size>::type> constexpr const Constant< Type, Size > **ratio** (const Type num, const Type den, const Types &...values) const

Start ratio multiplication.
- template<typename... Types, class = typename std::enable_if<sizeof...(Types) == Size>::type> constexpr const Constant< Type, Size > **ratio** (const Type, const Type, const Types...values) const

Stop ratio multiplication.

- template<typename... Types, class = typename std::enable_if<sizeof...(Types) != Size>::type>
constexpr const Constant< Type,
Size > **pow** (const int n, const Types &...values) const

Start power calculation.

- template<typename... Types, class = typename std::enable_if<sizeof...(Types) == Size>::type>
constexpr const Constant< Type,
Size > **pow** (const int, const Types...values) const

Stop power calculation.

- template<typename... Types, class = typename std::enable_if<sizeof...(Types) != Size>::type>
constexpr const Constant< Type,
Size > **irt** (const int n, const Types &...values) const

Start integer part root calculation.

- template<typename... Types, class = typename std::enable_if<sizeof...(Types) == Size>::type>
constexpr const Constant< Type,
Size > **irt** (const int, const Types...values) const

Stop integer root calculation.

- template<typename... Types, class = typename std::enable_if<sizeof...(Types) != Size>::type>
constexpr const Constant< Type,
Size > **ilog** (const int b, const Types &...values) const

Start integer logarithm calculation.

- template<typename... Types, class = typename std::enable_if<sizeof...(Types) == Size>::type>
constexpr const Constant< Type,
Size > **ilog** (const int, const Types...values) const

Stop integer logarithm calculation.

Static Public Member Functions

Helpers

- template<typename OtherType , unsigned int OtherSize, class Array , typename... Types, class = typename std::enable_if<sizeof...(-Types) != OtherSize>::type>
static constexpr const
std::array< OtherType,
OtherSize > **transmute** (const Array source, const Types &...values)
- Start conversion of array data types.*
- template<typename OtherType , unsigned int OtherSize, class Array , typename... Types, class = typename std::enable_if<sizeof...(-Types) == OtherSize>::type>
static constexpr const
std::array< OtherType,
OtherSize > **transmute** (const Array, const Types...values)
- Stop conversion of array data types.*
- template<typename OtherType , class = typename std::enable_if<std::is_arithmetic<OtherType>::value>::type>
static constexpr OtherType **metapow** (const OtherType x, const int n)
- Compile-time exponentiation.*
- template<typename OtherType , typename IntegralType = std::true_type, class = typename std::enable_if<(std::is_arithmetic<-OtherType>::value) && ((std::is_same<IntegralType, std::true_type>::value) || (std::is_integral<IntegralType>::value))>::type>
static constexpr OtherType **metairt** (const OtherType x, const int n, const IntegralType low=IntegralType(),
const IntegralType high=IntegralType(), const IntegralType mid=IntegralType())
- Compile-time integer root.*
- template<typename OtherType , class = typename std::enable_if<std::is_arithmetic<OtherType>::value>::type>
static constexpr OtherType **metailog** (const OtherType x, const int b)
- Compile-time integer logarithm.*

Test

- static int **example** ()

Example function.

Protected Attributes

Data members

- const std::array< Type, Size > `_data`
Set of numerical constants.

Friends

Stream

- template<typename SelfType , unsigned int SelfSize>
`std::ostream & operator<< (std::ostream &lhs, const Constant< SelfType, SelfSize > &rhs)`
Output stream operator.

6.15.1 Detailed Description

`template<typename Type = double, unsigned int Size = 1>exception magrathea::Constant< Type, Size >`

Numerical constant with `constexpr` constructor.

Provides a class for integral and numerical constants based on the structure of `std::integral_constant` with some extra mathematical functions. It accepts single or a set of constants internally stored in an array. As all is done at compile-time, all members works with copies of arithmetic values instead of references. Finally, one should define a constant to the maximum precision and calls the `cast` function to convert to a smaller precision.

Template Parameters

Type	Numerical type of the constant, that should have a <code>constexpr</code> constructor.
Size	Size of the set of constants, which is equal to one per default for a single value.

6.15.2 Constructor & Destructor Documentation

6.15.2.1 `template<typename Type , unsigned int Size> constexpr magrathea::Constant< Type, Size >::Constant (const Type source)`

Implicit `constexpr` value constructor.

Implicitly constructs the constant from a single constant arithmetic value of the same type.

Parameters

in	source	Single constant arithmetic value.
----	--------	-----------------------------------

6.15.2.2 `template<typename Type , unsigned int Size> template<typename... Types, class > constexpr magrathea::Constant< Type, Size >::Constant (const Types... source)`

Initial value:

```
{
    static_assert(1 == Size, "ERROR = Constant::Constant() : not compatible
size")
```

Implicit `constexpr` values constructor.

Implicitly constructs the constant from a set of constant arithmetic values.

Template Parameters

<i>Types</i>	(Variadic types.)
--------------	-------------------

Parameters

<i>in</i>	<i>source</i>	Set of constant arithmetic values.
-----------	---------------	------------------------------------

6.15.2.3 template<typename Type , unsigned int Size> template<typename OtherType, class > constexpr
magrathea::Constant< Type, Size >::Constant (const OtherType *source*)

Initial value:

```
{
    static_assert(sizeof...(source) == Size, "ERROR = Constant::Constant() :  
        not compatible size")
```

Implicit constexpr value conversion constructor.

Implicitly constructs the constant from a single constant arithmetic value of another type.

Template Parameters

<i>OtherType</i>	(Other data type.)
------------------	--------------------

Parameters

<i>in</i>	<i>source</i>	Single constant arithmetic value.
-----------	---------------	-----------------------------------

6.15.2.4 template<typename Type , unsigned int Size> template<typename OtherType, class > constexpr
magrathea::Constant< Type, Size >::Constant (const Constant< OtherType, Size > *source*)

Initial value:

```
{
    static_assert(1 == Size, "ERROR = Constant::Constant() : not compatible  
        size")
```

Implicit constexpr constant conversion constructor.

Implicitly constructs the constant from a constant of another type.

Template Parameters

<i>OtherType</i>	(Other data type.)
------------------	--------------------

Parameters

<i>in</i>	<i>source</i>	Other constant.
-----------	---------------	-----------------

6.15.2.5 template<typename Type , unsigned int Size> template<typename OtherType, class > constexpr
magrathea::Constant< Type, Size >::Constant (const std::array< OtherType, Size > *source*)

Implicit constexpr array conversion constructor.

Implicitly constructs the constant from a constant array of another type.

Template Parameters

<i>OtherType</i>	(Other data type.)
------------------	--------------------

Parameters

in	<i>source</i>	Array.
----	---------------	--------

6.15.2.6 `template<typename Type , unsigned int Size> constexpr magrathea::Constant< Type, Size >::Constant (const std::array< Type, Size > source)`

Implicit constexpr array constructor.

Implicitly constructs the constant from a constant array of the same type.

Parameters

in	<i>source</i>	Array.
----	---------------	--------

6.15.3 Member Function Documentation

6.15.3.1 `template<typename Type , unsigned int Size> template<typename... Types, class > constexpr const Constant< Type, Size > magrathea::Constant< Type, Size >::abs (const Types &... values) const`

Start calculation of the absolute value.

Starts recursive calculation of the absolute constant value : $|x|$.

Template Parameters

<i>Types</i>	(Variadic types.)
--------------	-------------------

Parameters

in	<i>values</i>	(Recursively extracted values.)
----	---------------	---------------------------------

Returns

Copy.

6.15.3.2 `template<typename Type , unsigned int Size> template<typename... Types, class > constexpr const Constant< Type, Size > magrathea::Constant< Type, Size >::abs (const Types... values) const`

Stop calculation of the absolute value.

Stops recursive calculation of the absolute constant value : $|x|$.

Template Parameters

<i>Types</i>	(Variadic types.)
--------------	-------------------

Parameters

in	<i>values</i>	(Recursively extracted values.)
----	---------------	---------------------------------

Returns

Copy.

6.15.3.3 template<typename Type , unsigned int Size> template<typename... Types, class > constexpr const Constant< Type, Size > magrathea::Constant< Type, Size >::add (const Type *rhs*, const Types &... *values*) const

Start addition.

Starts recursive addition of a value : $x + y$.

Template Parameters

Types	(Variadic types.)
--------------	-------------------

Parameters

in	<i>rhs</i>	Right-hand side.
in	<i>values</i>	(Recursively extracted values.)

Returns

Copy.

6.15.3.4 template<typename Type , unsigned int Size> template<typename... Types, class > constexpr const Constant< Type, Size > magrathea::Constant< Type, Size >::add (const Type , const Types... *values*) const

Stop addition.

Stops recursive addition of a value : $x + y$.

Template Parameters

Types	(Variadic types.)
--------------	-------------------

Parameters

in	<i>values</i>	(Recursively extracted values.)
----	---------------	---------------------------------

Returns

Copy.

6.15.3.5 template<typename Type , unsigned int Size> template<typename OtherType > constexpr const Constant< OtherType, Size > magrathea::Constant< Type, Size >::cast () const

Cast.

Casts to another constant type.

Template Parameters

OtherType	Other data type.
------------------	------------------

Returns

Copy.

6.15.3.6 template<typename Type , unsigned int Size> template<typename... Types, class > constexpr const Constant< Type, Size > magrathea::Constant< Type, Size >::cb (const Types &... values) const

Start calculation of the cube.

Starts recursive calculation of the cubed constant value : x^3 .

Template Parameters

<i>Types</i>	(Variadic types.)
--------------	-------------------

Parameters

<i>in</i>	<i>values</i>	(Recursively extracted values.)
-----------	---------------	---------------------------------

Returns

Copy.

6.15.3.7 template<typename Type , unsigned int Size> template<typename... Types, class > constexpr const Constant< Type, Size > magrathea::Constant< Type, Size >::cb (const Types... values) const

Stop calculation of the cube.

Stops recursive calculation of the cubed constant value : x^3 .

Template Parameters

<i>Types</i>	(Variadic types.)
--------------	-------------------

Parameters

<i>in</i>	<i>values</i>	(Recursively extracted values.)
-----------	---------------	---------------------------------

Returns

Copy.

6.15.3.8 template<typename Type , unsigned int Size> constexpr const Constant< Type, Size > magrathea::Constant< Type, Size >::copy () const

Copy.

Copies the constant.

Returns

Copy.

6.15.3.9 template<typename Type , unsigned int Size> template<typename OtherType > constexpr const std::array<OtherType, Size > **magrathea::Constant< Type, Size >::data() const**

Access data.

Returns the internal constant array casted to the desired type.

Template Parameters

<i>OtherType</i>	Other data type.
------------------	------------------

Returns

Copy of the array.

6.15.3.10 template<typename Type , unsigned int Size> template<typename... Types, class > constexpr const Constant< Type, Size > **magrathea::Constant< Type, Size >::div(const Type rhs, const Types &... values) const**

Start division.

Starts recursive division by a value : $\frac{x}{y}$.

Template Parameters

<i>Types</i>	(Variadic types.)
--------------	-------------------

Parameters

in	<i>rhs</i>	Right-hand side.
in	<i>values</i>	(Recursively extracted values.)

Returns

Copy.

6.15.3.11 template<typename Type , unsigned int Size> template<typename... Types, class > constexpr const Constant< Type, Size > **magrathea::Constant< Type, Size >::div(const Type , const Types... values) const**

Stop division.

Stops recursive division by a value : $\frac{x}{y}$.

Template Parameters

<i>Types</i>	(Variadic types.)
--------------	-------------------

Parameters

in	<i>values</i>	(Recursively extracted values.)
----	---------------	---------------------------------

Returns

Copy.

6.15.3.12 `template<typename Type , unsigned int Size> int magrathea::Constant< Type, Size >::example () [static]`

Example function.

Tests and demonstrates the use of [Constant](#).

Returns

0 if no error.

6.15.3.13 `template<typename Type , unsigned int Size> template<typename OtherType > constexpr OtherType magrathea::Constant< Type, Size >::get (const unsigned int i = 0) const`

Value getter.

Returns the specified value casted to the desired type.

Template Parameters

<i>OtherType</i>	Other data type.
------------------	------------------

Parameters

<i>in</i>	<i>i</i>	Index of the value.
-----------	----------	---------------------

Returns

Value.

6.15.3.14 `template<typename Type , unsigned int Size> template<typename... Types, class > constexpr const Constant< Type, Size > magrathea::Constant< Type, Size >::icbrt (const Types &... values) const`

Start calculation of the integer cube root.

Starts recursive calculation of the integer cube root constant value : $\left[\sqrt[3]{x} \right]$.

Template Parameters

<i>Types</i>	(Variadic types.)
--------------	-------------------

Parameters

<i>in</i>	<i>values</i>	(Recursively extracted values.)
-----------	---------------	---------------------------------

Returns

Copy.

6.15.3.15 `template<typename Type , unsigned int Size> template<typename... Types, class > constexpr const Constant< Type, Size > magrathea::Constant< Type, Size >::icbrt (const Types... values) const`

Stop calculation of the integer cube root.

Stops recursive calculation of the integer cube root constant value : $\left[\sqrt[3]{x} \right]$.

Template Parameters

<i>Types</i>	(Variadic types.)
--------------	-------------------

Parameters

<i>in</i>	<i>values</i>	(Recursively extracted values.)
-----------	---------------	---------------------------------

Returns

Copy.

6.15.3.16 template<typename Type , unsigned int Size> template<typename... Types, class > constexpr const Constant< Type, Size > magrathea::Constant< Type, Size >::ilog (const int *b*, const Types &... *values*) const

Start integer logarithm calculation.

Starts recursive integer logarithm calculation in the given base for integral arguments : $[\log_b([x])]$. For non-compatible arguments, the function returns 0.

Template Parameters

<i>Types</i>	(Variadic types.)
--------------	-------------------

Parameters

<i>in</i>	<i>b</i>	Logarithm integral base.
<i>in</i>	<i>values</i>	(Recursively extracted values.)

Returns

Copy.

6.15.3.17 template<typename Type , unsigned int Size> template<typename... Types, class > constexpr const Constant< Type, Size > magrathea::Constant< Type, Size >::ilog (const int , const Types... *values*) const

Stop integer logarithm calculation.

Stops recursive integer logarithm calculation in the given base for integral arguments : $[\log_b([x])]$. For non-compatible arguments, the function returns 0.

Template Parameters

<i>Types</i>	(Variadic types.)
--------------	-------------------

Parameters

<i>in</i>	<i>values</i>	(Recursively extracted values.)
-----------	---------------	---------------------------------

Returns

Copy.

6.15.3.18 `template<typename Type , unsigned int Size> template<typename... Types, class > constexpr const Constant< Type, Size > magrathea::Constant< Type, Size >::ilog10 (const Types &... values) const`

Start calculation of the integer base 10 logarithm.

Starts recursive calculation of the integer base 10 logarithm constant value : $[\log_{10} ([x])]$.

Template Parameters

<code>Types</code>	(Variadic types.)
--------------------	-------------------

Parameters

<code>in</code>	<code>values</code>	(Recursively extracted values.)
-----------------	---------------------	---------------------------------

Returns

Copy.

6.15.3.19 `template<typename Type , unsigned int Size> template<typename... Types, class > constexpr const Constant< Type, Size > magrathea::Constant< Type, Size >::ilog10 (const Types... values) const`

Stop calculation of the integer base 10 logarithm.

Stops recursive calculation of the integer base 10 logarithm constant value : $[\log_{10} ([x])]$.

Template Parameters

<code>Types</code>	(Variadic types.)
--------------------	-------------------

Parameters

<code>in</code>	<code>values</code>	(Recursively extracted values.)
-----------------	---------------------	---------------------------------

Returns

Copy.

6.15.3.20 `template<typename Type , unsigned int Size> template<typename... Types, class > constexpr const Constant< Type, Size > magrathea::Constant< Type, Size >::ilog2 (const Types &... values) const`

Start calculation of the integer base 2 logarithm.

Starts recursive calculation of the integer base 2 logarithm constant value : $[\log_2 ([x])]$.

Template Parameters

<code>Types</code>	(Variadic types.)
--------------------	-------------------

Parameters

<code>in</code>	<code>values</code>	(Recursively extracted values.)
-----------------	---------------------	---------------------------------

Returns

Copy.

6.15.3.21 template<typename Type , unsigned int Size> template<typename... Types, class > constexpr const Constant< Type, Size > magrathea::Constant< Type, Size >::ilog2(const Types... values) const

Stop calculation of the integer base 2 logarithm.

Stops recursive calculation of the integer base 2 logarithm constant value : $[\log_2([x])]$.

Template Parameters

<i>Types</i>	(Variadic types.)
--------------	-------------------

Parameters

<i>in</i>	<i>values</i>	(Recursively extracted values.)
-----------	---------------	---------------------------------

Returns

Copy.

6.15.3.22 template<typename Type , unsigned int Size> template<typename... Types, class > constexpr const Constant< Type, Size > magrathea::Constant< Type, Size >::inv(const Types &... values) const

Start calculation of the inverse.

Starts recursive calculation of the inverse constant value : $\frac{1}{x}$.

Template Parameters

<i>Types</i>	(Variadic types.)
--------------	-------------------

Parameters

<i>in</i>	<i>values</i>	(Recursively extracted values.)
-----------	---------------	---------------------------------

Returns

Copy.

6.15.3.23 template<typename Type , unsigned int Size> template<typename... Types, class > constexpr const Constant< Type, Size > magrathea::Constant< Type, Size >::inv(const Types... values) const

Stop calculation of the inverse.

Stops recursive calculation of the inverse constant value : $\frac{1}{x}$.

Template Parameters

<i>Types</i>	(Variadic types.)
--------------	-------------------

Parameters

in	<i>values</i>	(Recursively extracted values.)
----	---------------	---------------------------------

Returns

Copy.

6.15.3.24 template<typename Type , unsigned int Size> template<typename... Types, class > constexpr const Constant< Type, Size > magrathea::Constant< Type, Size >::irt (const int *n*, const Types &... *values*) const

Start integer part root calculation.

Starts recursive integer root calculation for integral arguments : $\sqrt[n]{[x]}$. For non-compatible arguments, the function returns 0.

Template Parameters

<i>Types</i>	(Variadic types.)
--------------	-------------------

Parameters

in	<i>n</i>	Root order.
in	<i>values</i>	(Recursively extracted values.)

Returns

Copy.

6.15.3.25 template<typename Type , unsigned int Size> template<typename... Types, class > constexpr const Constant< Type, Size > magrathea::Constant< Type, Size >::irt (const int , const Types... *values*) const

Stop integer root calculation.

Stops recursive integer root calculation for integral arguments : $\sqrt[n]{[x]}$. For non-compatible arguments, the function returns 0.

Template Parameters

<i>Types</i>	(Variadic types.)
--------------	-------------------

Parameters

in	<i>values</i>	(Recursively extracted values.)
----	---------------	---------------------------------

Returns

Copy.

6.15.3.26 template<typename Type , unsigned int Size> template<typename... Types, class > constexpr const Constant< Type, Size > magrathea::Constant< Type, Size >::isqrt (const Types &... *values*) const

Start calculation of the integer square root.

Starts recursive calculation of the integer square root constant value : $\sqrt{[x]}$.

Template Parameters

<i>Types</i>	(Variadic types.)
--------------	-------------------

Parameters

<i>in</i>	<i>values</i>	(Recursively extracted values.)
-----------	---------------	---------------------------------

Returns

Copy.

6.15.3.27 template<typename Type , unsigned int Size> template<typename... Types, class > constexpr const Constant< Type, Size > magrathea::Constant< Type, Size >::isqrt (const Types... values) const

Stop calculation of the integer square root.

Stops recursive calculation of the integer square root constant value : $\lceil \sqrt{[x]} \rceil$.

Template Parameters

<i>Types</i>	(Variadic types.)
--------------	-------------------

Parameters

<i>in</i>	<i>values</i>	(Recursively extracted values.)
-----------	---------------	---------------------------------

Returns

Copy.

6.15.3.28 template<typename Type , unsigned int Size> template<typename OtherType , class > constexpr OtherType magrathea::Constant< Type, Size >::metalog (const OtherType x, const int b) [static]

Compile-time integer logarithm.

Returns the value of the integer logarithm in the given base for integral arguments at compile-time : $[\log_b ([x])]$. For non-compatible arguments, the function returns 0.

Template Parameters

<i>OtherType</i>	(Other data type.)
------------------	--------------------

Parameters

<i>in</i>	<i>x</i>	Value.
<i>in</i>	<i>b</i>	Logarithm integral base.

Returns

Value of the result.

```
6.15.3.29 template<typename Type , unsigned int Size> template<typename OtherType , typename IntegralType , class >
constexpr OtherType magrathea::Constant< Type, Size >::metairt ( const OtherType x, const int n, const
IntegralType low = IntegralType (), const IntegralType high = IntegralType (), const IntegralType
mid = IntegralType () ) [static]
```

Compile-time integer root.

Returns the value of the integer root for integral arguments at compile-time : $\sqrt[n]{[x]}$. For non-compatible arguments, the function returns 0.

Template Parameters

<i>OtherType</i>	(Other data type.)
<i>IntegralType</i>	(Integer type.)

Parameters

in	<i>x</i>	Value.
in	<i>n</i>	Root order.
in	<i>low</i>	(Lower bound for binary recursive search.)
in	<i>high</i>	(Higher bound for binary recursive search.)
in	<i>mid</i>	(Mid value for binary recursive search.)

Returns

Value of the result.

```
6.15.3.30 template<typename Type , unsigned int Size> template<typename OtherType , class > constexpr OtherType
magrathea::Constant< Type, Size >::metapow ( const OtherType x, const int n ) [static]
```

Compile-time exponentiation.

Returns the value of the integral power of the argument : x^n . For negative exponents, the function returns the inverse of the integral power.

Template Parameters

<i>OtherType</i>	(Other data type.)
------------------	--------------------

Parameters

in	<i>x</i>	Value.
in	<i>n</i>	Exponent of the power.

Returns

Value of the result.

6.15.3.31 template<typename Type , unsigned int Size> template<typename... Types, class > constexpr const Constant< Type, Size > magrathea::Constant< Type, Size >::mod (const Type *rhs*, const Types &... *values*) const

Start modulo calculation.

Starts recursive modulo calculation : $x \% y$. For floating-point types, it computes the floating-point modulo.

Template Parameters

<i>Types</i>	(Variadic types.)
--------------	-------------------

Parameters

in	<i>rhs</i>	Right-hand side.
in	<i>values</i>	(Recursively extracted values.)

Returns

Copy.

6.15.3.32 template<typename Type , unsigned int Size> template<typename... Types, class > constexpr const Constant< Type, Size > magrathea::Constant< Type, Size >::mod (const Type , const Types... *values*) const

Stop modulo calculation.

Stops recursive modulo calculation : $x \% y$. For floating-point types, it computes the floating-point modulo.

Template Parameters

<i>Types</i>	(Variadic types.)
--------------	-------------------

Parameters

in	<i>values</i>	(Recursively extracted values.)
----	---------------	---------------------------------

Returns

Copy.

6.15.3.33 template<typename Type , unsigned int Size> template<typename... Types, class > constexpr const Constant< Type, Size > magrathea::Constant< Type, Size >::mul (const Type *rhs*, const Types &... *values*) const

Start multiplication.

Starts recursive multiplication by a value : $x \times y$.

Template Parameters

<i>Types</i>	(Variadic types.)
--------------	-------------------

Parameters

<i>in</i>	<i>rhs</i>	Right-hand side.
<i>in</i>	<i>values</i>	(Recursively extracted values.)

Returns

Copy.

6.15.3.34 template<typename Type , unsigned int Size> template<typename... Types, class > constexpr const Constant< Type, Size > magrathea::Constant< Type, Size >::mul (const Type , const Types... values) const

Stop multiplication.

Stops recursive multiplication by a value : $x \times y$.

Template Parameters

<i>Types</i>	(Variadic types.)
--------------	-------------------

Parameters

<i>in</i>	<i>values</i>	(Recursively extracted values.)
-----------	---------------	---------------------------------

Returns

Copy.

6.15.3.35 template<typename Type , unsigned int Size> constexpr const Constant< Type, Size > magrathea::Constant< Type, Size >::nullify () const

Nullify.

Returns a copy of the constant filled with null values.

Returns

Copy.

6.15.3.36 template<typename Type , unsigned int Size> constexpr magrathea::Constant< Type, Size >::operator Type () const

Value cast operator.

Implicitly returns the first value of the underlying array of constants. This should be used with caution in case of a set of constants as it only operates on the first element.

Returns

Value.

6.15.3.37 template<typename Type , unsigned int Size> template<typename... Types, class > constexpr const Constant< Type, Size > magrathea::Constant< Type, Size >::operator() (const Types &... values) const

Incomplete setter operator.

Starts recursive extraction of the missing values from the current array.

Template Parameters

<i>Types</i>	(Variadic types.)
--------------	-------------------

Parameters

<i>in</i>	<i>values</i>	Set of constant arithmetic values.
-----------	---------------	------------------------------------

Returns

Copy.

6.15.3.38 template<typename Type , unsigned int Size> template<typename... Types, class > constexpr const Constant< Type, Size > magrathea::Constant< Type, Size >::operator() (const Types... values) const

Setter operator.

Sets the values.

Template Parameters

<i>Types</i>	(Variadic types.)
--------------	-------------------

Parameters

<i>in</i>	<i>values</i>	Set of constant arithmetic values.
-----------	---------------	------------------------------------

Returns

Copy.

6.15.3.39 template<typename Type , unsigned int Size> constexpr Type magrathea::Constant< Type, Size >::operator() () const

Default getter operator.

Returns the first value.

Returns

Value.

6.15.3.40 template<typename Type , unsigned int Size> constexpr Type magrathea::Constant< Type, Size >::operator[] (const unsigned int i) const

Subscript operator.

Returns the specified value of the underlying array.

Parameters

<i>in</i>	<i>i</i>	Index of the value.
-----------	----------	---------------------

Returns

Value.

6.15.3.41 `template<typename Type , unsigned int Size> template<typename... Types, class > constexpr const Constant< Type, Size > magrathea::Constant< Type, Size >::opp (const Types &... values) const`

Start calculation of the opposite.

Starts recursive calculation of the opposite constant value : $-x$.

Template Parameters

<i>Types</i>	(Variadic types.)
--------------	-------------------

Parameters

<i>in</i>	<i>values</i>	(Recursively extracted values.)
-----------	---------------	---------------------------------

Returns

Copy.

6.15.3.42 `template<typename Type , unsigned int Size> template<typename... Types, class > constexpr const Constant< Type, Size > magrathea::Constant< Type, Size >::opp (const Types... values) const`

Stop calculation of opposite.

Stops recursive calculation of the opposite constant value : $-x$.

Template Parameters

<i>Types</i>	(Variadic types.)
--------------	-------------------

Parameters

<i>in</i>	<i>values</i>	(Recursively extracted values.)
-----------	---------------	---------------------------------

Returns

Copy.

6.15.3.43 `template<typename Type , unsigned int Size> template<typename... Types, class > constexpr const Constant< Type, Size > magrathea::Constant< Type, Size >::pow (const int n, const Types &... values) const`

Start power calculation.

Starts recursive power calculation for integral exponent : x^n . For negative exponents, the function returns the inverse of the integral power.

Template Parameters

<i>Types</i>	(Variadic types.)
--------------	-------------------

Parameters

in	<i>n</i>	Exponent of the power.
in	<i>values</i>	(Recursively extracted values.)

Returns

Copy.

6.15.3.44 template<typename Type , unsigned int Size> template<typename... Types, class > constexpr const Constant< Type, Size > **magrathea::Constant< Type, Size >::pow** (const int , const Types... *values*) const

Stop power calculation.

Stops recursive power calculation for integral exponent : x^n . For negative exponents, the function returns the inverse of the integral power.

Template Parameters

Types	(Variadic types.)
-------	-------------------

Parameters

in	<i>values</i>	(Recursively extracted values.)
----	---------------	---------------------------------

Returns

Copy.

6.15.3.45 template<typename Type , unsigned int Size> template<typename... Types, class > constexpr const Constant< Type, Size > **magrathea::Constant< Type, Size >::ratio** (const Type *num*, const Type *den*, const Types &... *values*) const

Start ratio multiplication.

Starts recursive ratio multiplication : $\frac{x \times num}{den}$.

Template Parameters

Types	(Variadic types.)
-------	-------------------

Parameters

in	<i>num</i>	Numerator.
in	<i>den</i>	Denominator.
in	<i>values</i>	(Recursively extracted values.)

Returns

Copy.

6.15.3.46 `template<typename Type , unsigned int Size> template<typename... Types, class > constexpr const Constant< Type, Size > magrathea::Constant< Type, Size >::ratio (const Type , const Type , const Types... values) const`

Stop ratio multiplication.

Stops recursive ratio multiplication : $\frac{x \times num}{den}$.

Template Parameters

<code>Types</code>	(Variadic types.)
--------------------	-------------------

Parameters

<code>in</code>	<code>values</code>	(Recursively extracted values.)
-----------------	---------------------	---------------------------------

Returns

Copy.

6.15.3.47 `template<typename Type , unsigned int Size> template<unsigned int OtherSize> constexpr const Constant< Type, OtherSize > magrathea::Constant< Type, Size >::resize () const`

Resize.

Returns a copy of the constant with a new size.

Returns

Copy.

6.15.3.48 `template<typename Type , unsigned int Size> template<typename... Types, class > constexpr const Constant< Type, Size > magrathea::Constant< Type, Size >::set (const Types &... values) const`

Incomplete value setter.

Starts recursive extraction of the missing values from the current array.

Template Parameters

<code>Types</code>	(Variadic types.)
--------------------	-------------------

Parameters

<code>in</code>	<code>values</code>	Set of constant arithmetic values.
-----------------	---------------------	------------------------------------

Returns

Copy.

6.15.3.49 template<typename Type , unsigned int Size> template<typename... Types, class > constexpr const Constant< Type, Size > **magrathea::Constant< Type, Size >::set** (const Types... values) const

Value setter.

Sets the values.

Template Parameters

Types	(Variadic types.)
--------------	-------------------

Parameters

in	values	Set of constant arithmetic values.
-----------	---------------	------------------------------------

Returns

Copy.

6.15.3.50 template<typename Type , unsigned int Size> template<typename... Types, class > constexpr const Constant< Type, Size > **magrathea::Constant< Type, Size >::sgn** (const Types &... values) const

Start calculation of the the signum function.

Starts recursive calculation of the signum constant value : $sgn(x = 0) = 0, sgn(x < 0) = -1, sgn(x > 0) = 1$.

Template Parameters

Types	(Variadic types.)
--------------	-------------------

Parameters

in	values	(Recursively extracted values.)
-----------	---------------	---------------------------------

Returns

Copy.

6.15.3.51 template<typename Type , unsigned int Size> template<typename... Types, class > constexpr const Constant< Type, Size > **magrathea::Constant< Type, Size >::sgn** (const Types... values) const

Stop calculation of the the signum function.

Stops recursive calculation of the signum constant value : $sgn(x = 0) = 0, sgn(x < 0) = -1, sgn(x > 0) = 1$.

Template Parameters

Types	(Variadic types.)
--------------	-------------------

Parameters

in	values	(Recursively extracted values.)
-----------	---------------	---------------------------------

Returns

Copy.

6.15.3.52 `template<typename Type , unsigned int Size> template<class Ratio , typename... Types, class > constexpr const Constant< Type, Size > magrathea::Constant< Type, Size >::si(const Types &... values) const`

Start multiplication by a SI prefix.

Starts recursive multiplication by a SI prefix ratio : $\frac{x \times num}{den}$.

Template Parameters

<i>Ratio</i>	Standard ratio type.
<i>Types</i>	(Variadic types.)

Parameters

<i>in</i>	<i>values</i>	(Recursively extracted values.)
-----------	---------------	---------------------------------

Returns

Copy.

6.15.3.53 `template<typename Type , unsigned int Size> template<class Ratio , typename... Types, class > constexpr const Constant< Type, Size > magrathea::Constant< Type, Size >::si(const Types... values) const`

Stop multiplication by a SI prefix.

Stops recursive multiplication by a SI prefix ratio : $\frac{x \times num}{den}$.

Template Parameters

<i>Ratio</i>	Standard ratio type.
<i>Types</i>	(Variadic types.)

Parameters

<i>in</i>	<i>values</i>	(Recursively extracted values.)
-----------	---------------	---------------------------------

Returns

Copy.

6.15.3.54 `template<typename Type , unsigned int Size> constexpr unsigned int magrathea::Constant< Type, Size >::size() const`

Size.

Returns the size of the constant.

Returns

Value of the size.

6.15.3.55 `template<typename Type , unsigned int Size> template<typename... Types, class > constexpr const Constant< Type, Size > magrathea::Constant< Type, Size >::sq(const Types &... values) const`

Start calculation of the square.

Starts recursive calculation of the squared constant value : x^2 .

Template Parameters

<i>Types</i>	(Variadic types.)
--------------	-------------------

Parameters

<i>in</i>	<i>values</i>	(Recursively extracted values.)
-----------	---------------	---------------------------------

Returns

Copy.

6.15.3.56 template<typename Type , unsigned int Size> template<typename... Types, class > constexpr const Constant< Type, Size > magrathea::Constant< Type, Size >::sq (const Types... values) const

Stop calculation of the square.

Stops recursive calculation of the squared constant value : x^2 .

Template Parameters

<i>Types</i>	(Variadic types.)
--------------	-------------------

Parameters

<i>in</i>	<i>values</i>	(Recursively extracted values.)
-----------	---------------	---------------------------------

Returns

Copy.

6.15.3.57 template<typename Type , unsigned int Size> template<typename... Types, class > constexpr const Constant< Type, Size > magrathea::Constant< Type, Size >::sub (const Type rhs, const Types &... values) const

Start substraction.

Starts recursive substraction of a value : $x - y$.

Template Parameters

<i>Types</i>	(Variadic types.)
--------------	-------------------

Parameters

<i>in</i>	<i>rhs</i>	Right-hand side.
<i>in</i>	<i>values</i>	(Recursively extracted values.)

Returns

Copy.

6.15.3.58 template<typename Type , unsigned int Size> template<typename... Types, class > constexpr const Constant< Type, Size > magrathea::Constant< Type, Size >::sub (const Type , const Types... values) const

Stop substraction.

Stops recursive subtraction of a value : $x - y$.

Template Parameters

<i>Types</i>	(Variadic types.)
--------------	-------------------

Parameters

<i>in</i>	<i>values</i>	(Recursively extracted values.)
-----------	---------------	---------------------------------

Returns

Copy.

6.15.3.59 template<typename Type , unsigned int Size> template<typename OtherType , unsigned int OtherSize, class Array , typename... Types, class > constexpr const std::array< OtherType, OtherSize > magrathea::Constant< Type, Size >::transmute (const Array source, const Types &... values) [static]

Start conversion of array data types.

Recursively extracts and casts data from source to construct an array of another type.

Template Parameters

<i>OtherType</i>	Other data type.
<i>OtherSize</i>	Other size.
<i>Array</i>	(Array type.)
<i>Types</i>	(Variadic types.)

Parameters

<i>in</i>	<i>source</i>	Array to convert.
<i>in</i>	<i>values</i>	(Recursively extracted values.)

Returns

Casted array.

6.15.3.60 template<typename Type , unsigned int Size> template<typename OtherType , unsigned int OtherSize, class Array , typename... Types, class > constexpr const std::array< OtherType, OtherSize > magrathea::Constant< Type, Size >::transmute (const Array , const Types... values) [static]

Stop conversion of array data types.

Stops the recursive extraction and constructs the final array from data.

Template Parameters

<i>OtherType</i>	Other data type.
<i>OtherSize</i>	Other size.
<i>Array</i>	(Array type.)
<i>Types</i>	(Variadic types.)

Parameters

<code>in</code>	<code>values</code>	(Recursively extracted values.)
-----------------	---------------------	---------------------------------

Returns

Casted array.

6.15.3.61 template<typename Type , unsigned int Size> template<typename OtherType > constexpr OtherType magrathea::Constant< Type, Size >::value (const unsigned int *i* = 0) const

Access a value.

Returns the specified value casted to the desired type.

Template Parameters

<code>OtherType</code>	Other data type.
------------------------	------------------

Parameters

<code>in</code>	<code>i</code>	Index of the value.
-----------------	----------------	---------------------

Returns

Value.

6.15.4 Friends And Related Function Documentation

6.15.4.1 template<typename Type = double, unsigned int Size = 1> template<typename SelfType , unsigned int SelfSize> std::ostream& operator<< (std::ostream & *lhs*, const Constant< SelfType, SelfSize > & *rhs*) [friend]

[Output](#) stream operator.

Adds each element to the stream using the `fill()` character as a separator.

Template Parameters

<code>SelfType</code>	(Numerical type of the constant.)
<code>SelfSize</code>	(Size of the set of constants.)

Parameters

<code>in,out</code>	<code>lhs</code>	Left-hand side stream.
<code>in</code>	<code>rhs</code>	Right-hand side constant.

Returns

[Output](#) stream.

6.15.5 Member Data Documentation

6.15.5.1 template<typename Type = double, unsigned int Size = 1> const std::array<Type, Size> magrathea::Constant< Type, Size >::_data [protected]

Set of numerical constants.

The documentation for this exception was generated from the following file:

- /data/home/mbreton/magrathea_pathfinder/src/magrathea/constant.h

6.16 magrathea::Constants< Type > Exception Template Reference

Common mathematical and physical constants.

```
#include <constants.h>
```

Static Public Member Functions

Mathematical constants

- static constexpr const
`Constant< Type > pi ()`
Pi.
- static constexpr const
`Constant< Type > napier ()`
Napier constant.
- static constexpr const
`Constant< Type > euler ()`
Euler-Mascheroni constant.
- static constexpr const
`Constant< Type > golden ()`
Golden ratio.
- static constexpr const
`Constant< Type > pythagoras ()`
Pythagoras constant, square root of 2.
- static constexpr const
`Constant< Type > theodorus ()`
Theodorus constant, square root of 3.
- static constexpr const
`Constant< Type > glaisher ()`
Glaisher-Kinkelin constant.
- static constexpr const
`Constant< Type > khinchin ()`
Khinchin constant.
- static constexpr const
`Constant< Type > soldner ()`
Ramanujan-Soldner constant.
- static constexpr const
`Constant< Type > mertens ()`
Meissel-Mertens constant.
- static constexpr const
`Constant< Type > plastic ()`
Plastic number.
- static constexpr const
`Constant< Type > catalan ()`
Catalan constant.
- static constexpr const
`Constant< Type > feigenbaumd ()`
First Feigenbaum constant.
- static constexpr const
`Constant< Type > feigenbauma ()`
Second Feigenbaum constant.

- static constexpr const
Constant< Type > omega ()
Omega constant.
- static constexpr const
Constant< Type > tanarosec ()
Value of the tangent of one arcsecond.
- static constexpr const
Constant< Type > sqrtpi ()
Square root of pi.
- static constexpr const
Constant< Type > pi2 ()
Squared pi.

Universal constants

- static constexpr const
Constant< Type > c ()
Speed of light in vacuum.
- static constexpr const
Constant< Type > g ()
Newtonian constant of gravitation.
- static constexpr const
Constant< Type > h ()
Planck constant.
- static constexpr const
Constant< Type > hbar ()
Reduced Planck constant.
- static constexpr const
Constant< Type > c2 ()
Squared speed of light in vacuum.
- static constexpr const
Constant< Type > g2 ()
Squared newtonian constant of gravitation.
- static constexpr const
Constant< Type > h2 ()
Squared Planck constant.
- static constexpr const
Constant< Type > hbar2 ()
Squared reduced Planck constant.
- static constexpr const
Constant< Type > c4 ()
Speed of light in vacuum to the fourth.

Units

- static constexpr const
Constant< Type > celsius ()
Value of zero celsius degree in kelvin.
- static constexpr const
Constant< Type > atomic ()
Value of the atomic mass in kilograms.
- static constexpr const
Constant< Type > atm ()
Value of one atmosphere in pascals.
- static constexpr const
Constant< Type > gn ()
Acceleration of gravity in meters per second squared.
- static constexpr const
Constant< Type > au ()

- static constexpr const
Constant< Type > ev ()
Value of an astronomical unit in meters.
- static constexpr const
Constant< Type > evc2 ()
Value of an electron-volt in joules.
- static constexpr const
Constant< Type > deg ()
Value of a degree in radians.
- static constexpr const
Constant< Type > arcmin ()
Value of an arcminute in radians.
- static constexpr const
Constant< Type > arcsec ()
Value of an arcsecond in radians.
- static constexpr const
Constant< Type > deg2 ()
Value of a square degree in steradians.
- static constexpr const
Constant< Type > arcmin2 ()
Value of a square arcminute in steradians.
- static constexpr const
Constant< Type > arcsec2 ()
Value of a square arcsecond in steradians.
- static constexpr const
Constant< Type > sphere ()
Value of a sphere in steradians.
- static constexpr const
Constant< Type > second ()
Duration of a second in seconds.
- static constexpr const
Constant< Type > minute ()
Duration of a minute in seconds.
- static constexpr const
Constant< Type > hour ()
Duration of an hour in seconds.
- static constexpr const
Constant< Type > day ()
Duration of a SI julian day in seconds.
- static constexpr const
Constant< Type > year ()
Duration of a SI julian year in seconds.
- static constexpr const
Constant< Type > ly ()
Value of a light-year in meters.
- static constexpr const
Constant< Type > pc ()
Value of a parsec in meters.

Particle masses

- static constexpr const
Constant< Type > mquarku ()
Up quark mass.
- static constexpr const
Constant< Type > mquarkd ()
Down quark mass.

- static constexpr const
`Constant< Type > mquarkc ()`
Charm quark mass.
- static constexpr const
`Constant< Type > mquarks ()`
Strange quark mass.
- static constexpr const
`Constant< Type > mquarkt ()`
Top quark mass.
- static constexpr const
`Constant< Type > mquarkb ()`
Bottom quark mass.
- static constexpr const
`Constant< Type > melectron ()`
Electron mass.
- static constexpr const
`Constant< Type > mmuon ()`
Muon mass.
- static constexpr const
`Constant< Type > mtau ()`
Tau mass.
- static constexpr const
`Constant< Type > mnuelectron ()`
Electron neutrino mass.
- static constexpr const
`Constant< Type > mnumuon ()`
Muon neutrino mass.
- static constexpr const
`Constant< Type > mnutau ()`
Tau neutrino mass.
- static constexpr const
`Constant< Type > mphoton ()`
Photon mass.
- static constexpr const
`Constant< Type > mbosonw ()`
W boson mass.
- static constexpr const
`Constant< Type > mbosonz ()`
Z boson mass.
- static constexpr const
`Constant< Type > mgluon ()`
Gluon mass.
- static constexpr const
`Constant< Type > mhiggs ()`
Higgs boson mass.
- static constexpr const
`Constant< Type > mgraviton ()`
Graviton mass.
- static constexpr const
`Constant< Type > mproton ()`
Proton mass.
- static constexpr const
`Constant< Type > mneutron ()`
Neutron mass.

Electromagnetic constants

- static constexpr const
`Constant< Type > fcd ()`

- static constexpr const
Constant< Type > kcd ()
Luminous efficacy used in candela definition.
- static constexpr const
Constant< Type > e ()
Elementary charge.
- static constexpr const
Constant< Type > e2 ()
Squared elementary charge.
- static constexpr const
Constant< Type > mu0 ()
Vacuum permeability.
- static constexpr const
Constant< Type > epsilon0 ()
Vacuum permittivity.
- static constexpr const
Constant< Type > z0 ()
Characteristic impedance of vacuum.
- static constexpr const
Constant< Type > ke ()
Coulomb's constant.
- static constexpr const
Constant< Type > magnetonb ()
Bohr magneton.
- static constexpr const
Constant< Type > magnetonn ()
Nuclear magneton.
- static constexpr const
Constant< Type > quantumc ()
Quantum conductance.
- static constexpr const
Constant< Type > quantumf ()
Quantum flux.
- static constexpr const
Constant< Type > josephson ()
Josephson constant.
- static constexpr const
Constant< Type > klitzing ()
Von Klitzing constant.

Physico-chemical constants

- static constexpr const
Constant< Type > na ()
Avogadro number.
- static constexpr const
Constant< Type > wien ()
Wien constant.
- static constexpr const
Constant< Type > kb ()
Boltzmann constant.
- static constexpr const
Constant< Type > kb2 ()
Squared boltzmann constant.
- static constexpr const
Constant< Type > gas ()
Ideal gas constant.

- static constexpr const
Constant< Type > radiationf ()
First radiation constant.
- static constexpr const
Constant< Type > radiations ()
Second radiation constant.
- static constexpr const
Constant< Type > faraday ()
Faraday constant.
- static constexpr const
Constant< Type > stefan ()
Stefan constant.

Atomic and nuclear constants

- static constexpr const
Constant< Type > hfs ()
Ground state hyperfine splitting frequency of caesium 133.
- static constexpr const
Constant< Type > alpha ()
Fine structure constant.
- static constexpr const
Constant< Type > alpha2 ()
Squared fine structure constant.
- static constexpr const
Constant< Type > rydberg ()
Rydberg constant.
- static constexpr const
Constant< Type > relectron ()
Classical electron radius.
- static constexpr const
Constant< Type > rbohr ()
Bohr radius.
- static constexpr const
Constant< Type > hartree ()
Hartree energy.
- static constexpr const
Constant< Type > thomson ()
Thomson cross section.

Planck units

- static constexpr const
Constant< Type > planckq ()
Planck charge.
- static constexpr const
Constant< Type > planckl ()
Planck length.
- static constexpr const
Constant< Type > planckm ()
Planck mass.
- static constexpr const
Constant< Type > planckt ()
Planck time.
- static constexpr const
Constant< Type > planckf ()
Planck force.
- static constexpr const
Constant< Type > plancke ()

- static constexpr const
Constant< Type > planckp ()
Planck power.
- static constexpr const
Constant< Type > plancktheta ()
Planck temperature.

Solar system masses

- static constexpr const
Constant< Type > msun ()
Solar mass.
- static constexpr const
Constant< Type > mearth ()
Earth mass.
- static constexpr const
Constant< Type > mmmercury ()
Mercury mass.
- static constexpr const
Constant< Type > mvenus ()
Venus mass.
- static constexpr const
Constant< Type > mmars ()
Mars mass.
- static constexpr const
Constant< Type > mjupiter ()
Jupiter mass.
- static constexpr const
Constant< Type > msaturn ()
Saturn mass.
- static constexpr const
Constant< Type > muranus ()
Uranus mass.
- static constexpr const
Constant< Type > mneptune ()
Neptune mass.
- static constexpr const
Constant< Type > mmoon ()
Moon mass.

Solar system radii

- static constexpr const
Constant< Type > rsun ()
Solar radius.
- static constexpr const
Constant< Type > rearth ()
Venus radius.
- static constexpr const
Constant< Type > rmercury ()
Earth radius.
- static constexpr const
Constant< Type > rvenus ()
Mercury radius.
- static constexpr const
Constant< Type > rmars ()
Mars radius.

- static constexpr const
`Constant< Type > rjupiter ()`
Jupiter radius.
- static constexpr const
`Constant< Type > rsaturn ()`
Saturn radius.
- static constexpr const
`Constant< Type > ruranus ()`
Uranus radius.
- static constexpr const
`Constant< Type > rneptune ()`
Neptune radius.
- static constexpr const
`Constant< Type > rmoon ()`
Moon radius.

Test

- static int `example ()`
Example function.

6.16.1 Detailed Description

`template<typename Type = double>exception magrathea::Constants< Type >`

Common mathematical and physical constants.

Provides an unified access to common mathematical and physical constants in SI units. When the square of the constant is widely used in common formulas, it is provided too. The values were taken in 2012 from :

- Wikipedia : http://en.wikipedia.org/wiki/Mathematical_constant http://en.wikipedia.org/wiki/Physical_constants
- CODATA : <http://www.codata.org>
- Particle Data Group : <http://pdg.lbl.gov>
- International Astronomical Union : <http://www.iau.org>

Template Parameters

Type	Type of the constants.
------	------------------------

6.16.2 Member Function Documentation

6.16.2.1 template<typename Type > constexpr const Constant< Type > magrathea::Constants< Type >::alpha() [static]

Fine structure constant.

Gets the value of α .

Returns

α .

6.16.2.2 template<typename Type > constexpr const Constant< Type > magrathea::Constants< Type >::alpha2() [static]

Squared fine structure constant.

Gets the value of α^2 .

Returns

α^2 .

6.16.2.3 template<typename Type > constexpr const Constant< Type > magrathea::Constants< Type >::arcmin() [static]

Value of an arcminute in radians.

Gets the value of $arcmin$.

Returns

$arcmin$.

6.16.2.4 template<typename Type > constexpr const Constant< Type > magrathea::Constants< Type >::arcmin2() [static]

Value of a square arcminute in steradians.

Gets the value of $arcmin^2$.

Returns

$arcmin^2$.

6.16.2.5 template<typename Type > constexpr const Constant< Type > magrathea::Constants< Type >::arcsec() [static]

Value of an arcsecond in radians.

Gets the value of $arcsec$.

Returns

$arcsec$.

6.16.2.6 template<typename Type > constexpr const Constant< Type > magrathea::Constants< Type >::arcsec2() [static]

Value of a square arcsecond in steradians.

Gets the value of $arcsec^2$.

Returns

$arcsec^2$.

6.16.2.7 template<typename Type> constexpr const Constant<Type> **magrathea::Constants<Type>::atm**()
[static]

Value of one atmosphere in pascals.

Gets the value of *atm*.

Returns

atm.

6.16.2.8 template<typename Type> constexpr const Constant<Type> **magrathea::Constants<Type>::atomic**()
[static]

Value of the atomic mass in kilograms.

Gets the value of *u*.

Returns

u.

6.16.2.9 template<typename Type> constexpr const Constant<Type> **magrathea::Constants<Type>::au**()
[static]

Value of an astronomical unit in meters.

Gets the value of *au*.

Returns

au.

6.16.2.10 template<typename Type> constexpr const Constant<Type> **magrathea::Constants<Type>::c**()
[static]

Speed of light in vacuum.

Gets the value of *c*.

Returns

c.

6.16.2.11 template<typename Type> constexpr const Constant<Type> **magrathea::Constants<Type>::c2**()
[static]

Squared speed of light in vacuum.

Gets the value of c^2 .

Returns

c^2 .

6.16.2.12 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::c4() [static]`

Speed of light in vacuum to the fourth.

Gets the value of c^4 .

Returns

c^4 .

6.16.2.13 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::catalan() [static]`

Catalan constant.

Gets the value of K .

Returns

K .

6.16.2.14 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::celsius() [static]`

Value of zero celsius degree in kelvin.

Gets the value of ${}^\circ C$.

Returns

${}^\circ C$.

6.16.2.15 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::day() [static]`

Duration of a SI julian day in seconds.

Gets the value of d .

Returns

d .

6.16.2.16 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::deg() [static]`

Value of a degree in radians.

Gets the value of deg .

Returns

deg .

6.16.2.17 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::deg2() [static]`

Value of a square degree in steradians.

Gets the value of deg^2 .

Returns

deg^2 .

6.16.2.18 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::e() [static]`

Elementary charge.

Gets the value of e .

Returns

e .

6.16.2.19 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::e2() [static]`

Squared elementary charge.

Gets the value of e^2 .

Returns

e^2 .

6.16.2.20 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::epsilon0() [static]`

Vacuum permittivity.

Gets the value of ϵ_0 .

Returns

ϵ_0 .

6.16.2.21 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::euler() [static]`

Euler-Mascheroni constant.

Gets the value of γ .

Returns

γ .

6.16.2.22 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::ev() [static]`

Value of an electron-volt in joules.

Gets the value of eV .

Returns

eV .

6.16.2.23 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::evc2() [static]`

Value of the electron-volt mass unit in kilograms.

Gets the value of $\frac{eV}{c^2}$.

Returns

$\frac{eV}{c^2}$.

6.16.2.24 `template<typename Type> int magrathea::Constants<Type>::example() [static]`

Example function.

Tests and demonstrates the use of `Constants`.

Returns

0 if no error.

6.16.2.25 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::faraday() [static]`

Faraday constant.

Gets the value of F .

Returns

F .

6.16.2.26 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::fcd() [static]`

Frequency of radiation used in candela definition.

Gets the value of v_{cd} .

Returns

v_{cd} .

```
6.16.2.27 template<typename Type> constexpr const Constant< Type > magrathea::Constants< Type >::feigenbaum( ) [static]
```

Second Feigenbaum constant.

Gets the value of α .

Returns

α .

```
6.16.2.28 template<typename Type> constexpr const Constant< Type > magrathea::Constants< Type >::feigenbaumd( ) [static]
```

First Feigenbaum constant.

Gets the value of δ .

Returns

δ .

```
6.16.2.29 template<typename Type> constexpr const Constant< Type > magrathea::Constants< Type >::g( ) [static]
```

Newtonian constant of gravitation.

Gets the value of G .

Returns

G .

```
6.16.2.30 template<typename Type> constexpr const Constant< Type > magrathea::Constants< Type >::g2( ) [static]
```

Squared newtonian constant of gravitation.

Gets the value of g^2 .

Returns

g^2 .

```
6.16.2.31 template<typename Type> constexpr const Constant< Type > magrathea::Constants< Type >::gas( ) [static]
```

Ideal gas constant.

Gets the value of R .

Returns

R .

6.16.2.32 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::glaisher() [static]`

Glaisher-Kinkelin constant.

Gets the value of A .

Returns

A .

6.16.2.33 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::gn() [static]`

Acceleration of gravity in meters per second squared.

Gets the value of g_n .

Returns

g_n .

6.16.2.34 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::golden() [static]`

Golden ratio.

Gets the value of $\varphi = \frac{1+\sqrt{5}}{2}$.

Returns

$\varphi = \frac{1+\sqrt{5}}{2}$.

6.16.2.35 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::h() [static]`

Planck constant.

Gets the value of h .

Returns

h .

6.16.2.36 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::h2() [static]`

Squared Planck constant.

Gets the value of h^2 .

Returns

h^2 .

6.16.2.37 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::hartree() [static]`

Hartree energy.

Gets the value of E_h .

Returns

E_h .

6.16.2.38 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::hbar() [static]`

Reduced Planck constant.

Gets the value of \bar{h} .

Returns

\bar{h} .

6.16.2.39 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::hbar2() [static]`

Squared reduced Planck constant.

Gets the value of \bar{h}^2 .

Returns

\bar{h}^2 .

6.16.2.40 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::hfs() [static]`

Ground state hyperfine splitting frequency of caesium 133.

Gets the value of $\Delta\nu(^{133}\text{Cs})_{hfs}$.

Returns

$\Delta\nu(^{133}\text{Cs})_{hfs}$.

6.16.2.41 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::hour() [static]`

Duration of an hour in seconds.

Gets the value of h .

Returns

h .

6.16.2.42 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::josephson() [static]`

Josephson constant.

Gets the value of K_J .

Returns

K_J .

6.16.2.43 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::kb() [static]`

Boltzmann constant.

Gets the value of k_B .

Returns

k_B .

6.16.2.44 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::kb2() [static]`

Squared boltzmann constant.

Gets the value of k_B^2 .

Returns

k_B^2 .

6.16.2.45 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::kcd() [static]`

Luminous efficacy used in candela definition.

Gets the value of K_{cd} .

Returns

K_{cd} .

6.16.2.46 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::ke() [static]`

Coulomb's constant.

Gets the value of k_e .

Returns

k_e .

6.16.2.47 template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::khinchin() [static]

Khinchin constant.

Gets the value of K_0 .

Returns

K_0 .

6.16.2.48 template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::klitzing() [static]

Von Klitzing constant.

Gets the value of R_K .

Returns

R_K .

6.16.2.49 template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::ly() [static]

Value of a light-year in meters.

Gets the value of ly .

Returns

ly .

6.16.2.50 template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::magnetonb() [static]

Bohr magneton.

Gets the value of μ_B .

Returns

μ_B .

6.16.2.51 template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::magnetonn() [static]

Nuclear magneton.

Gets the value of μ_N .

Returns

μ_N .

6.16.2.52 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::mbosonw() [static]`

W boson mass.

Gets the value of m_W .

Returns

m_W .

6.16.2.53 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::mbosonz() [static]`

Z boson mass.

Gets the value of m_Z .

Returns

m_Z .

6.16.2.54 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::mearth() [static]`

Earth mass.

Gets the value of m_E .

Returns

m_E .

6.16.2.55 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::melectron() [static]`

Electron mass.

Gets the value of m_e .

Returns

m_e .

6.16.2.56 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::mertens() [static]`

Meissel-Mertens constant.

Gets the value of M_1 .

Returns

M_1 .

6.16.2.57 `template<typename Type> constexpr const Constant< Type > magrathea::Constants< Type >::mgluon() [static]`

Gluon mass.

Gets the value of m_g .

Returns

m_g .

6.16.2.58 `template<typename Type> constexpr const Constant< Type > magrathea::Constants< Type >::mgraviton() [static]`

Graviton mass.

Gets the value of $m_{graviton}$.

Returns

$m_{graviton}$.

6.16.2.59 `template<typename Type> constexpr const Constant< Type > magrathea::Constants< Type >::mhiggs() [static]`

Higgs boson mass.

Gets the value of m_H .

Returns

m_H .

6.16.2.60 `template<typename Type> constexpr const Constant< Type > magrathea::Constants< Type >::minute() [static]`

Duration of a minute in seconds.

Gets the value of min .

Returns

min .

6.16.2.61 `template<typename Type> constexpr const Constant< Type > magrathea::Constants< Type >::mjupiter() [static]`

Jupiter mass.

Gets the value of m_J .

Returns

m_J .

6.16.2.62 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::mmars() [static]`

Mars mass.

Gets the value of m_{Ma} .

Returns

m_{Ma} .

6.16.2.63 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::mmercury() [static]`

Mercury mass.

Gets the value of m_{Me} .

Returns

m_{Me} .

6.16.2.64 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::mmoon() [static]`

Moon mass.

Gets the value of m_{Moon} .

Returns

m_{Moon} .

6.16.2.65 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::mmuon() [static]`

Muon mass.

Gets the value of m_μ .

Returns

m_μ .

6.16.2.66 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::mneptune() [static]`

Neptune mass.

Gets the value of m_N .

Returns

m_N .

6.16.2.67 template<typename Type> constexpr const Constant< Type > **magrathea::Constants< Type >::mneutron**() [static]

Neutron mass.

Gets the value of m_n .

Returns

m_n .

6.16.2.68 template<typename Type> constexpr const Constant< Type > **magrathea::Constants< Type >::mnuelectron**() [static]

Electron neutrino mass.

Gets the value of m_{ν_e} .

Returns

m_{ν_e} .

6.16.2.69 template<typename Type> constexpr const Constant< Type > **magrathea::Constants< Type >::mnumuon**() [static]

Muon neutrino mass.

Gets the value of m_{ν_μ} .

Returns

m_{ν_μ} .

6.16.2.70 template<typename Type> constexpr const Constant< Type > **magrathea::Constants< Type >::mnutau**() [static]

Tau neutrino mass.

Gets the value of m_{ν_τ} .

Returns

m_{ν_τ} .

6.16.2.71 template<typename Type> constexpr const Constant< Type > **magrathea::Constants< Type >::mphoton**() [static]

Photon mass.

Gets the value of m_γ .

Returns

m_γ .

6.16.2.72 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::mproton() [static]`

Proton mass.

Gets the value of m_p .

Returns

m_p .

6.16.2.73 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::mquarkb() [static]`

Bottom quark mass.

Gets the value of m_b .

Returns

m_b .

6.16.2.74 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::mquarkc() [static]`

Charm quark mass.

Gets the value of m_c .

Returns

m_c .

6.16.2.75 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::mquarkd() [static]`

Down quark mass.

Gets the value of m_d .

Returns

m_d .

6.16.2.76 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::mquarks() [static]`

Strange quark mass.

Gets the value of m_s .

Returns

m_s .

6.16.2.77 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::mquarkt() [static]`

Top quark mass.

Gets the value of m_t .

Returns

m_t .

6.16.2.78 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::mquarku() [static]`

Up quark mass.

Gets the value of m_u .

Returns

m_u .

6.16.2.79 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::msaturn() [static]`

Saturn mass.

Gets the value of m_{Sa} .

Returns

m_{Sa} .

6.16.2.80 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::msun() [static]`

Solar mass.

Gets the value of m_{\odot} .

Returns

m_{\odot} .

6.16.2.81 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::mtau() [static]`

Tau mass.

Gets the value of m_{τ} .

Returns

m_{τ} .

6.16.2.82 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::mu0() [static]`

Vacuum permeability.

Gets the value of μ_0 .

Returns

μ_0 .

6.16.2.83 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::muranus() [static]`

Uranus mass.

Gets the value of m_U .

Returns

m_U .

6.16.2.84 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::mvenus() [static]`

Venus mass.

Gets the value of m_{Ve} .

Returns

m_{Ve} .

6.16.2.85 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::na() [static]`

Avogadro number.

Gets the value of N_A .

Returns

N_A .

6.16.2.86 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::napier() [static]`

Napier constant.

Gets the value of e .

Returns

e .

6.16.2.87 template<typename Type> constexpr const Constant<Type> **magrathea::Constants< Type >::omega**()
[static]

Omega constant.

Gets the value of Ω .

Returns

Ω .

6.16.2.88 template<typename Type> constexpr const Constant<Type> **magrathea::Constants< Type >::pc**()
[static]

Value of a parsec in meters.

Gets the value of pc .

Returns

pc .

6.16.2.89 template<typename Type> constexpr const Constant<Type> **magrathea::Constants< Type >::pi**()
[static]

Pi.

Gets the value of π .

Returns

π .

6.16.2.90 template<typename Type> constexpr const Constant<Type> **magrathea::Constants< Type >::pi2**()
[static]

Squared pi.

Gets the value of π^2 .

Returns

π^2 .

6.16.2.91 template<typename Type> constexpr const Constant<Type> **magrathea::Constants< Type >::plancke**()
[static]

Planck energy.

Gets the value of E_P .

Returns

E_P .

6.16.2.92 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::planckf() [static]`

Planck force.

Gets the value of F_P .

Returns

F_P .

6.16.2.93 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::plancki() [static]`

Planck length.

Gets the value of l_P .

Returns

l_P .

6.16.2.94 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::planckm() [static]`

Planck mass.

Gets the value of m_P .

Returns

m_P .

6.16.2.95 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::planckp() [static]`

Planck power.

Gets the value of P_P .

Returns

P_P .

6.16.2.96 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::planckq() [static]`

Planck charge.

Gets the value of q_P .

Returns

q_P .

6.16.2.97 template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::planckt()
[static]

Planck time.

Gets the value of t_P .

Returns

t_P .

6.16.2.98 template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::plancktheta()
[static]

Planck temperature.

Gets the value of θ_P .

Returns

θ_P .

6.16.2.99 template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::plastic()
[static]

Plastic number.

Gets the value of ρ .

Returns

ρ .

6.16.2.100 template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::pythagoras()
[static]

Pythagoras constant, square root of 2.

Gets the value of $\sqrt{2}$.

Returns

$\sqrt{2}$.

6.16.2.101 template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::quantumc()
[static]

Quantum conductance.

Gets the value of G_0 .

Returns

G_0 .

6.16.2.102 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::quantumf() [static]`

Quantum flux.

Gets the value of ϕ_0 .

Returns

ϕ_0 .

6.16.2.103 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::radiationf() [static]`

First radiation constant.

Gets the value of c_1 .

Returns

c_1 .

6.16.2.104 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::radiations() [static]`

Second radiation constant.

Gets the value of c_2 .

Returns

c_2 .

6.16.2.105 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::rbohr() [static]`

Bohr radius.

Gets the value of a_0 .

Returns

a_0 .

6.16.2.106 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::rearth() [static]`

Venus radius.

Gets the value of r_{Ve} .

Returns

r_{Ve} .

6.16.2.107 template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::relectron()
[static]

Classical electron radius.

Gets the value of r_e .

Returns

r_e .

6.16.2.108 template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::rjupiter()
[static]

Jupiter radius.

Gets the value of r_J .

Returns

r_J .

6.16.2.109 template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::rmars()
[static]

Mars radius.

Gets the value of r_{Ma} .

Returns

r_{Ma} .

6.16.2.110 template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::rmercury()
[static]

Earth radius.

Gets the value of r_E .

Returns

r_E .

6.16.2.111 template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::rmoon()
[static]

Moon radius.

Gets the value of r_{Moon} .

Returns

r_{Moon} .

6.16.2.112 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::rneptune() [static]`

Neptune radius.

Gets the value of r_N .

Returns

r_N .

6.16.2.113 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::rsaturn() [static]`

Saturn radius.

Gets the value of r_{Sa} .

Returns

r_{Sa} .

6.16.2.114 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::rsun() [static]`

Solar radius.

Gets the value of r_\odot .

Returns

r_\odot .

6.16.2.115 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::ruranus() [static]`

Uranus radius.

Gets the value of r_U .

Returns

r_U .

6.16.2.116 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::rvenus() [static]`

Mercury radius.

Gets the value of r_{Me} .

Returns

r_{Me} .

6.16.2.117 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::rydberg() [static]`

Rydberg constant.

Gets the value of R_∞ .

Returns

R_∞ .

6.16.2.118 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::second() [static]`

Duration of a second in seconds.

Gets the value of s .

Returns

s .

6.16.2.119 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::soldner() [static]`

Ramanujan-Soldner constant.

Gets the value of μ .

Returns

μ .

6.16.2.120 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::sphere() [static]`

Value of a sphere in steradians.

Gets the value of $4\pi sr$.

Returns

$4\pi sr$.

6.16.2.121 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::sqrtpi() [static]`

Square root of pi.

Gets the value of $\sqrt{\pi}$.

Returns

$\sqrt{\pi}$.

6.16.2.122 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::stefan() [static]`

Stefan constant.

Gets the value of σ .

Returns

σ .

6.16.2.123 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::tanarcsec() [static]`

Value of the tangent of one arcsecond.

Gets the value of $\tan(\text{arcsec})$.

Returns

$\tan(\text{arcsec})$.

6.16.2.124 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::theodorus() [static]`

Theodorus constant, square root of 3.

Gets the value of $\sqrt{3}$.

Returns

$\sqrt{3}$.

6.16.2.125 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::thomson() [static]`

Thomson cross section.

Gets the value of σ_T .

Returns

σ_T .

6.16.2.126 `template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::wien() [static]`

Wien constant.

Gets the value of b .

Returns

b .

6.16.2.127 template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::year()
[static]

Duration of a SI julian year in seconds.

Gets the value of *year*.

Returns

year.

6.16.2.128 template<typename Type> constexpr const Constant<Type> magrathea::Constants<Type>::z0()
[static]

Characteristic impedance of vacuum.

Gets the value of Z_0 .

Returns

Z_0 .

The documentation for this exception was generated from the following file:

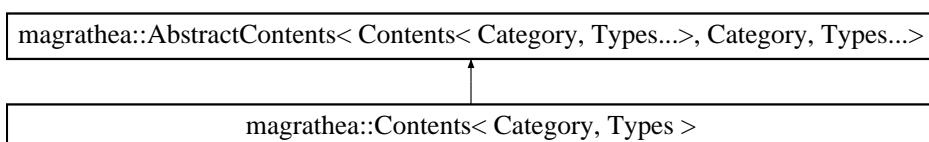
- /data/home/mbreton/magrathea_pathfinder/src/magrathea/constants.h

6.17 magrathea::Contents< Category, Types > Exception Template Reference

Basic implementation of numerical simulation contents.

```
#include <contents.h>
```

Inheritance diagram for magrathea::Contents< Category, Types >:



Public Member Functions

Lifecycle

- template<class... Misc>
[Contents](#) (Misc &&...misc)
Explicit generic constructor.

Static Public Member Functions

Test

- static int [example](#) ()
Example function.

Public Attributes

- using `operator =` = `typedef`

Additional Inherited Members

6.17.1 Detailed Description

`template<class Category = void, class... Types>exception magrathea::Contents< Category, Types >`

Basic implementation of numerical simulation contents.

This class is the direct derivation of `AbstractContents`. It provides the most basic and generic contents object without adding new functionalities to the abstract class. It can be used in most cases as a generic container of groups of physical quantities.

Template Parameters

<code>Category</code>	<code>Contents</code> category (Lagrangian, Eulerian, Grid...).
<code>Types</code>	Variadic list of components types.

6.17.2 Constructor & Destructor Documentation

6.17.2.1 `template<class Category , class... Types> template<class... Misc> magrathea::Contents< Category, Types >::Contents(Misc &&... misc) [inline], [explicit]`

Explicit generic constructor.

Provides a generic interface to all constructors of the base class.

Template Parameters

<code>Misc</code>	(Miscellaneous types.)
-------------------	------------------------

Parameters

<code>in</code>	<code>misc</code>	Miscellaneous arguments.
-----------------	-------------------	--------------------------

6.17.3 Member Function Documentation

6.17.3.1 `template<class Category , class... Types> int magrathea::Contents< Category, Types >::example() [static]`

Example function.

Tests and demonstrates the use of `Contents`.

Returns

0 if no error.

6.17.4 Member Data Documentation

6.17.4.1 `template<class Category = void, class... Types> using magrathea::Contents< Category, Types >::operator =`

The documentation for this exception was generated from the following file:

- /data/home/mbreton/magrathea_pathfinder/src/magrathea/contents.h

6.18 Create_octree Class Reference

```
#include <create_octree.h>
```

Public Member Functions

- template<template< typename Type, class Index, class Data, unsigned int Dimension, class Position, class Extent, class Element, class Container > class Octree, typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container , class Parameter , class Cone , class FileList , class Sphere , typename Integer , typename Scalar > void [PreparationHDF5_from_particles](#) (Octree< Type, Index, Data, Dimension, Position, Extent, Element, Container > &octree, const Parameter &[parameters](#), const Integer ntasks, const Integer rank, const std::vector< Cone > &cone, const std::vector< Cone > &coneflRot, const Scalar thetay, const Scalar thetaz, FileList &conefile, const Sphere µsphere)
- Create Octree binary files.*
- template<template< typename Type, class Index, class Data, unsigned int Dimension, class Position, class Extent, class Element, class Container > class Octree, typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container , class Parameter , class Cone , class Sphere , class FileList , class Index2 , typename Integer , typename Scalar , class Cosmology > void [PreparationBinary](#) (Octree< Type, Index, Data, Dimension, Position, Extent, Element, Container > &octree, const Parameter &[parameters](#), const Integer ntasks, const Integer rank, const std::vector< Cone > &cone, FileList &conefile, const Sphere µsphere, Index2 &filetree, const Scalar h, const Scalar omegam, const Scalar lboxmpch, Scalar &amin, const Cosmology &cosmology)
- Create Octree binary files.*
- template<template< typename Type, class Index, class Data, unsigned int Dimension, class Position, class Extent, class Element, class Container > class Octree, typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container , class Parameter , class Cone , class FileList , class Sphere , typename Integer , typename Scalar , class Cosmology > void [PreparationASCII](#) (Octree< Type, Index, Data, Dimension, Position, Extent, Element, Container > &octree, const Parameter &[parameters](#), const Integer ntasks, const Integer rank, const std::vector< Cone > &cone, const std::vector< Cone > &coneflRot, const std::array< std::array< double, 3 >, 3 > &rotm1, FileList &conefile, const Sphere µsphere, const Scalar h, const Scalar omegam, const Scalar lboxmpch, Scalar &amin, const Cosmology &cosmology)

Create Octree binary files.

Static Public Member Functions

- template<class Parameters , class Map > static void [ReadParamFile](#) (Parameters &[parameters](#), Map ¶meter)
- Read parameter file.*
- template<template< typename Type, class Index, class Data, unsigned int Dimension, class Position, class Extent, class Element, class Container > class Octree, typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container , class Parameters , class Cone , class FileList , class Sphere , typename Integer , typename Scalar , class Cosmology > static void [PreparationHDF5_from_cells](#) (Octree< Type, Index, Data, Dimension, Position, Extent, Element, Container > &octree, const Parameters &[parameters](#), const Integer ntasks, const Integer rank, const std::vector< Cone > &cone, const std::vector< Cone > &coneflRot, const std::array< std::array< double, 3 >, 3 > &rotm1, const Scalar thetay, const Scalar thetaz, FileList &conefile, const Sphere µsphere, const Scalar h, const Scalar omegam, const Scalar lboxmpch, Scalar &amin, const Cosmology &cosmology)
- Create Octree binary files.*
- template<template< typename Type, class Index, class Data, unsigned int Dimension, class Position, class Extent, class Element, class Container > class Octree, typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container , class Parameters , class Cone , class FileList , class Sphere , typename Integer , typename Scalar >

```

static void PreparationHDF5_from_particles (Octree< Type, Index, Data, Dimension, Position, Extent, Element, Container > &octree, const Parameters &parameters, const Integer ntasks, const Integer rank, const std::vector< Cone > &cone, const std::vector< Cone > &coneflRot, const Scalar thetay, const Scalar thetaz, FileList &conefile, const Sphere &microsphere)

• template<template< typename Type, class Index, class Data, unsigned int Dimension, class Position, class Extent, class Element, class Container > class Octree, typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container , class Parameters , class Cone , class Sphere , class FileList , class Index2 , typename Integer , typename Scalar , class Cosmology >

static void PreparationBinary (Octree< Type, Index, Data, Dimension, Position, Extent, Element, Container > &octree, const Parameters &parameters, const Integer ntasks, const Integer rank, const std::vector< Cone > &cone, FileList &conefile, const Sphere &microsphere, Index2 &filetree, const Scalar h, const Scalar omegam, const Scalar lboxmpch, Scalar &amin, const Cosmology &cosmology)

• template<template< typename Type, class Index, class Data, unsigned int Dimension, class Position, class Extent, class Element, class Container > class Octree, typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container , class Parameters , class Cone , class FileList , class Sphere , typename Integer , typename Scalar , class Cosmology >

static void PreparationASCII (Octree< Type, Index, Data, Dimension, Position, Extent, Element, Container > &octree, const Parameters &parameters, const Integer ntasks, const Integer rank, const std::vector< Cone > &cone, const std::vector< Cone > &coneflRot, const std::array< std::array< double, 3 >, 3 > &rotm1, FileList &conefile, const Sphere &microsphere, const Scalar h, const Scalar omegam, const Scalar lboxmpch, Scalar &amin, const Cosmology &cosmology)

• template<class Parameters , typename Type1 , template< typename Type, class Index, class Data, unsigned int Dimension, class Position, class Extent, class Element, class Container > class Octree, typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container >

static void CreateOctreeWithCIC (bool &continue_refining, Octree< Type, Index, Data, Dimension, Position, Extent, Element, Container > &octree, const Parameters &parameters, std::vector< Type1 > &pos_part, std::vector< Type1 > &force_part, std::vector< Type1 > &potential_part, std::vector< Type1 > &a_part)

    Create Octree using CIC.

• template<class Parameters , typename Type1 , template< typename Type, class Index, class Data, unsigned int Dimension, class Position, class Extent, class Element, class Container > class Octree, typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container >

static void CreateOctreeWithTSC (bool &continue_refining, Octree< Type, Index, Data, Dimension, Position, Extent, Element, Container > &octree, const Parameters &parameters, std::vector< Type1 > &pos_part, std::vector< Type1 > &force_part, std::vector< Type1 > &potential_part, std::vector< Type1 > &a_part)

    Create Octree using TSC.

```

6.18.1 Member Function Documentation

6.18.1.1 template<class Parameters , typename Type1 , template< typename Type, class Index, class Data, unsigned int Dimension, class Position, class Extent, class Element, class Container > class Octree, typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > void CreateOctreeWithCIC (bool & continue_refining, Octree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & octree, const Parameters & parameters, std::vector< Type1 > & pos_part, std::vector< Type1 > & force_part, std::vector< Type1 > & potential_part, std::vector< Type1 > & a_part)
[static]

Create Octree using CIC.

Create Octree using CIC from particles

Template Parameters

<i>Parameters</i>	Parameters type
<i>Type1</i>	Type1 type
<i>Octree</i>	octree type
<i>Type</i>	type type
<i>Index</i>	index type

<i>Data</i>	data type
<i>Dimension</i>	Number of dimensions
<i>Position</i>	position type
<i>Extent</i>	extent type
<i>Element</i>	element type
<i>Container</i>	container type

Parameters

in,out	<i>continue_refining</i>	Boolean which controls if we continue refinement
in,out	<i>octree</i>	Octree be to filled using particles
in	<i>parameters</i>	Parameters structure
in	<i>pos_part</i>	Position of particles
in	<i>force_part</i>	Force of particles
in	<i>potential_part</i>	Potential of particles
in	<i>a_part</i>	Scale factor of particles

```
6.18.1.2 template<class Parameters , typename Type1 , template< typename Type, class Index, class Data, unsigned int Dimension, class Position, class Extent, class Element, class Container > class Octree, typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > void Create_octree::CreateOctreeWithTSC ( bool & continue_refining, Octree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & octree, const Parameters & parameters, std::vector< Type1 > & pos_part, std::vector< Type1 > & force_part, std::vector< Type1 > & potential_part, std::vector< Type1 > & a_part ) [static]
```

Create Octree using TSC.

Create Octree using TSC

Template Parameters

<i>Parameters</i>	Parameters type
<i>Type1</i>	Type1 type
<i>Octree</i>	octree type
<i>Type</i>	type type
<i>Index</i>	index type
<i>Data</i>	data type
<i>Dimension</i>	Number of dimensions
<i>Position</i>	position type
<i>Extent</i>	extent type
<i>Element</i>	element type
<i>Container</i>	container type

Parameters

in,out	<i>continue_refining</i>	Boolean which controls if we continue refinement
in,out	<i>octree</i>	Octree be to filled using particles
in	<i>parameters</i>	Parameters structure
in	<i>pos_part</i>	Position of particles
in	<i>force_part</i>	Force of particles
in	<i>potential_part</i>	Potential of particles
in	<i>a_part</i>	Scale factor of particles

6.18.1.3 template<template< typename Type, class Index, class Data, unsigned int Dimension, class Position, class Extent, class Element, class Container > class Octree, typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container , class Parameters , class Cone , class FileList , class Sphere , typename Integer , typename Scalar , class Cosmology > static void Create_octree::PreparationASCII (Octree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & octree, const Parameters & parameters, const Integer ntasks, const Integer rank, const std::vector< Cone > & cone, const std::vector< Cone > & coneflRot, const std::array< std::array< double, 3 >, 3 > & rotm1, FileList & coneFile, const Sphere & microsphere, const Scalar h, const Scalar omegam, const Scalar lboxmpch, Scalar & amin, const Cosmology & cosmology) [static]

6.18.1.4 template<template< typename Type, class Index, class Data, unsigned int Dimension, class Position, class Extent, class Element, class Container > class Octree, typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container , class Parameter , class Cone , class FileList , class Sphere , typename Integer , typename Scalar , class Cosmology > void Create_octree::PreparationASCII (Octree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & octree, const Parameter & parameters, const Integer ntasks, const Integer rank, const std::vector< Cone > & cone, const std::vector< Cone > & coneflRot, const std::array< std::array< double, 3 >, 3 > & rotm1, FileList & coneFile, const Sphere & microsphere, const Scalar h, const Scalar omegam, const Scalar lboxmpch, Scalar & amin, const Cosmology & cosmology)

Create Octree binary files.

Create Octree binary files from ASCII files.

Template Parameters

<i>Octree</i>	octree type
<i>Type</i>	type type
<i>Index</i>	index type
<i>Data</i>	data type
<i>Dimension</i>	Number of dimensions
<i>Position</i>	position type
<i>Extent</i>	extent type
<i>Element</i>	element type
<i>Container</i>	container type
<i>Parameters</i>	Parameters type
<i>Cone</i>	cone type
<i>Filelist</i>	filelist type
<i>Sphere</i>	sphere type
<i>Integer</i>	Integer type
<i>Scalar</i>	scalar type
<i>Cosmology</i>	cosmology type

Parameters

in,out	<i>octree</i>	Octree to be filled
in	<i>parameters</i>	Parameters structure
in	<i>ntasks</i>	Number of tasks
in	<i>rank</i>	Process rank
in	<i>cone</i>	<i>Cone</i> properties
in	<i>coneflRot</i>	<i>Cone</i> with possibly rotation
in	<i>rotm1</i>	Rotation matrix for narrow cells
in	<i>coneFile</i>	<i>Cone</i> names in conedir
in	<i>microsphere</i>	Central buffer zone for Octree
in	<i>h</i>	dimensionless Hubble parameter
in	<i>omegam</i>	Matter density fraction
in	<i>lboxmpch</i>	Size of simulation box
in	<i>amin</i>	minimum value scale factor
in	<i>cosmology</i>	Cosmological tables

- 6.18.1.5 template<template< typename Type, class Index, class Data, unsigned int Dimension, class Position, class Extent, class Element, class Container > class Octree, typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container , class Parameters , class Cone , class Sphere , class FileList , class Index2 , typename Integer , typename Scalar , class Cosmology > static void Create_octree::PreparationBinary (Octree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & octree, const Parameters & parameters, const Integer ntasks, const Integer rank, const std::vector< Cone > & cone, FileList & cone file, const Sphere & microsphere, Index2 & filer tree, const Scalar h, const Scalar omegam, const Scalar lboxmpch, Scalar & amin, const Cosmology & cosmology) [static]
- 6.18.1.6 template<template< typename Type, class Index, class Data, unsigned int Dimension, class Position, class Extent, class Element, class Container > class Octree, typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container , class Parameter , class Cone , class Sphere , class FileList , class Index2 , typename Integer , typename Scalar , class Cosmology > void Create_octree::PreparationBinary (Octree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & octree, const Parameter & parameters, const Integer ntasks, const Integer rank, const std::vector< Cone > & cone, FileList & cone file, const Sphere & microsphere, Index2 & filer tree, const Scalar h, const Scalar omegam, const Scalar lboxmpch, Scalar & amin, const Cosmology & cosmology)

Create Octree binary files.

Create Octree binary files from Binary.

Template Parameters

<i>Octree</i>	octree type
<i>Type</i>	type type
<i>Index</i>	index type
<i>Data</i>	data type
<i>Dimension</i>	Number of dimensions
<i>Position</i>	position type
<i>Extent</i>	extent type
<i>Element</i>	element type
<i>Container</i>	container type
<i>Parameters</i>	Parameters type
<i>Cone</i>	cone type
<i>Filelist</i>	filelist type
<i>Sphere</i>	sphere type
<i>Integer</i>	Integer type
<i>Scalar</i>	scalar type
<i>Cosmology</i>	cosmology type

Parameters

in,out	<i>octree</i>	Octree to be filled
in	<i>parameters</i>	Parameters structure
in	<i>ntasks</i>	Number of tasks
in	<i>rank</i>	Process rank
in	<i>cone</i>	<i>Cone</i> properties
in	<i>cone file</i>	<i>Cone</i> names in conedir
in	<i>microsphere</i>	Central buffer zone for Octree
in	<i>filer tree</i>	Type of Octree containing strings
in	<i>h</i>	dimensionless Hubble parameter
in	<i>omegam</i>	Matter density fraction
in	<i>lboxmpch</i>	Size of simulation box
in	<i>amin</i>	minimum value scale factor
in	<i>cosmology</i>	Cosmological tables

```
6.18.1.7 template<template< typename Type, class Index, class Data, unsigned int Dimension, class Position, class Extent,
class Element, class Container > class Octree, typename Type , class Index , class Data , unsigned int Dimension, class
Position , class Extent , class Element , class Container , class Parameters , class Cone , class FileList , class Sphere ,
typename Integer , typename Scalar , class Cosmology > void Create_octree::PreparationHDF5_from_cells ( Octree<
Type, Index, Data, Dimension, Position, Extent, Element, Container > & octree, const Parameters & parameters, const
Integer ntasks, const Integer rank, const std::vector< Cone > & cone, const std::vector< Cone > & coneflRot,
const std::array< std::array< double, 3 >, 3 > & rotm1, const Scalar thetay, const Scalar thetaz, FileList &
conefile, const Sphere & microsphere, const Scalar h, const Scalar omegam, const Scalar lboxmpch, Scalar & amin,
const Cosmology & cosmology ) [static]
```

Create Octree binary files.

Create Octree binary files from HDF5.

Template Parameters

<i>Octree</i>	octree type
<i>Type</i>	type type
<i>Index</i>	index type
<i>Data</i>	data type
<i>Dimension</i>	Number of dimensions
<i>Position</i>	position type
<i>Extent</i>	extent type
<i>Element</i>	element type
<i>Container</i>	container type
<i>Parameters</i>	Parameters type
<i>Cone</i>	cone type
<i>Filelist</i>	filelist type
<i>Sphere</i>	sphere type
<i>Integer</i>	Integer type
<i>Scalar</i>	scalar type
<i>Cosmology</i>	cosmology type

Parameters

in,out	<i>octree</i>	Octree to be filled
in	<i>parameters</i>	Parameters structure
in	<i>ntasks</i>	Number of tasks
in	<i>rank</i>	Process rank
in	<i>cone</i>	<i>Cone</i> properties
in	<i>coneflRot</i>	<i>Cone</i> with possibly rotation
in	<i>rotm1</i>	Rotation matrix for narrow cells
in	<i>thetay</i>	Semi-angle for solid angle in direction y
in	<i>thetaz</i>	Semi-angle for solid angle in direction z
in	<i>conefile</i>	<i>Cone</i> names in conedir
in	<i>microsphere</i>	Central buffer zone for Octree
in	<i>h</i>	Dimensionless Hubble parameter
in	<i>omegam</i>	Matter density fraction
in	<i>lboxmpch</i>	Size of simulation box
in	<i>amin</i>	minimum value scale factor
in	<i>cosmology</i>	Cosmological tables

6.18.1.8 template<template< typename Type, class Index, class Data, unsigned int Dimension, class Position, class Extent, class Element, class Container > class Octree, typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container , class Parameters , class Cone , class FileList , class Sphere , typename Integer , typename Scalar > static void Create_octree::PreparationHDF5_from_particles (Octree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & octree, const Parameters & parameters, const Integer ntasks, const Integer rank, const std::vector< Cone > & cone, const std::vector< Cone > & coneflRot, const Scalar thetay, const Scalar thetz, FileList & conefile, const Sphere & microsphere) [static]

6.18.1.9 template<template< typename Type, class Index, class Data, unsigned int Dimension, class Position, class Extent, class Element, class Container > class Octree, typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container , class Parameter , class Cone , class FileList , class Sphere , typename Integer , typename Scalar > void Create_octree::PreparationHDF5_from_particles (Octree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & octree, const Parameter & parameters, const Integer ntasks, const Integer rank, const std::vector< Cone > & cone, const std::vector< Cone > & coneflRot, const Scalar thetay, const Scalar thetz, FileList & conefile, const Sphere & microsphere)

Create Octree binary files.

Create Octree binary files from HDF5.

Template Parameters

<i>Octree</i>	octree type
<i>Type</i>	type type
<i>Index</i>	index type
<i>Data</i>	data type
<i>Dimension</i>	Number of dimensions
<i>Position</i>	position type
<i>Extent</i>	extent type
<i>Element</i>	element type
<i>Container</i>	container type
<i>Parameters</i>	Parameters type
<i>Cone</i>	cone type
<i>Filelist</i>	filelist type
<i>Sphere</i>	sphere type
<i>Integer</i>	Integer type
<i>Scalar</i>	scalar type
<i>Cosmology</i>	cosmology type

Parameters

in,out	<i>octree</i>	Octree to be filled
in	<i>parameters</i>	Parameters structure
in	<i>ntasks</i>	Number of tasks
in	<i>rank</i>	Process rank
in	<i>cone</i>	<i>Cone</i> properties
in	<i>coneflRot</i>	<i>Cone</i> with possibly rotation
in	<i>conefile</i>	<i>Cone</i> names in conedir
in	<i>microsphere</i>	Central buffer zone for Octree
in	<i>h</i>	Dimensionless Hubble parameter
in	<i>omegam</i>	Matter density fraction
in	<i>lboxmpch</i>	Size of simulation box
in	<i>amin</i>	minimum value scale factor
in	<i>cosmology</i>	Cosmological tables

6.18.1.10 template<class Parameters , class Map > void CreateOctree::ReadParamFile (Parameters & parameters, Map & parameter) [static]

Read parameter file.

Read and put in a structure the parameters.

Template Parameters

<i>Parameters</i>	structure type
<i>Map</i>	map type

Parameters

<i>in,out</i>	<i>parameters</i>	Structure containing the parameters.
<i>in</i>	<i>parameter</i>	Contains parameters to be rewritten

The documentation for this class was generated from the following file:

- /data/home/mbreton/magrathea_pathfinder/src/[create_octree.h](#)

6.19 magrathea::DataHandler Exception Reference

Set of basic operations on binary data related to IO.

```
#include <datahandler.h>
```

Static Public Member Functions

Utilities

- template<typename Type >
static constexpr bool **array** (const Type &)
Is not an array.
- template<typename Type , std::size_t Size>
static constexpr bool **array** (const std::array< Type, Size > &)
Is an array.
- template<typename Type >
static constexpr bool **tuple** (const Type &)
Is not a tuple.
- template<typename... Types>
static constexpr bool **tuple** (const std::tuple< Types... > &)
Is a tuple.

Size

- template<typename... Types>
static constexpr unsigned int **size** ()
Number of elements of several types.
- static constexpr unsigned int **size** (const std::tuple<> &)
Number of elements of an empty tuple.
- template<typename Type >
static constexpr unsigned int **size** (const Type &variable)
Number of elements of a single variable.
- template<typename Type , typename... Types>
static constexpr unsigned int **size** (const Type &variable, const Types &...variables)
Number of elements of several variables.

- template<typename Type , std::size_t Size>
static constexpr unsigned int **size** (const std::array< Type, Size > &container)
Number of elements of an array.
- template<unsigned int Current = 0, typename... Types>
static constexpr unsigned int **size** (const std::tuple< Types...> &container)
Number of elements of a tuple.
- static unsigned long long int **rsize** (const std::nullptr_t=nullptr, const std::nullptr_t=nullptr)
Number of elements of an empty range.
- template<typename Type >
static unsigned long long int **rsize** (const Type *const first, const Type *const last)
Number of elements of a range between pointers.
- template<typename Type , class = typename std::enable_if<!std::is_pointer<Type>::value>::type>
static unsigned long long int **rsize** (const Type &first, const Type &last, typename std::iterator_traits< Type >::iterator_category *=nullptr)
Number of elements of a range between iterators.

Bytesize

- template<typename Type = void>
static constexpr unsigned int **bytesize** (const std::nullptr_t=nullptr)
Size in bytes of a single type.
- template<typename Type , typename... Types, class = typename std::enable_if<sizeof...(Types) != 0>::type>
static constexpr unsigned int **bytesize** ()
Size in bytes of a single type.
- static constexpr unsigned int **bytesize** (const std::tuple<> &)
Size in bytes of an empty tuple.
- template<typename Type >
static constexpr unsigned int **bytesize** (const Type &variable)
Size in bytes of a single variable.
- template<typename Type , typename... Types>
static constexpr unsigned int **bytesize** (const Type &variable, const Types &...variables)
Size in bytes of several variables.
- template<typename Type , std::size_t Size>
static constexpr unsigned int **bytesize** (const std::array< Type, Size > &container)
Size in bytes of an array.
- template<unsigned int Current = 0, typename... Types>
static constexpr unsigned int **bytesize** (const std::tuple< Types...> &container)
Size in bytes of a tuple.
- static unsigned long long int **rbytesize** (const std::nullptr_t=nullptr, const std::nullptr_t=nullptr)
Size in bytes of an empty range.
- template<typename Type >
static unsigned long long int **rbytesize** (const Type *const first, const Type *const last)
Size in bytes of a range between pointers.
- template<typename Type , class = typename std::enable_if<!std::is_pointer<Type>::value>::type>
static unsigned long long int **rbytesize** (const Type &first, const Type &last, typename std::iterator_traits< Type >::iterator_category *=nullptr)
Size in bytes of a range between iterators.

Byteswap

- template<bool Byteswap, typename... Types, class = typename std::enable_if<!Byteswap>::type>
static unsigned int **byteswap** (const Types &...)
Do not swap bytes.
- template<bool Byteswap, typename... Types, class = typename std::enable_if<!Byteswap>::type>
static unsigned int **rbyteswap** (const Types &...)
Do not swap bytes of a range.
- template<bool Byteswap = true, class = typename std::enable_if<Byteswap>::type>
static unsigned int **byteswap** ()

Swap bytes of nothing.

- template<bool Byteswap = true, class = typename std::enable_if<Byteswap>::type>
static unsigned int **byteswap** (std::tuple<> &)

Swap bytes of an empty tuple.

- template<bool Byteswap = true, typename Type , class = typename std::enable_if<Byteswap>::type>
static unsigned int **byteswap** (Type &variable)

Swap bytes of a single variable.

- template<bool Byteswap = true, typename Type , typename... Types, class = typename std::enable_if<Byteswap>::type>
static unsigned int **byteswap** (Type &variable, Types &...variables)

Swap bytes of several variables.

- template<bool Byteswap = true, typename Type , std::size_t Size, class = typename std::enable_if<Byteswap>::type>
static unsigned int **byteswap** (std::array< Type, Size > &container)

Swap bytes of an array.

- template<bool Byteswap = true, unsigned int Current = 0, typename... Types, class = typename std::enable_if<Byteswap>::type>
static unsigned int **byteswap** (std::tuple< Types...> &container)

Swap bytes of a tuple.

- template<bool Byteswap = true, class = typename std::enable_if<Byteswap>::type>
static unsigned long long int **rbyteswap** (const std::nullptr_t=nullptr, const std::nullptr_t=nullptr)

Swap bytes of an empty range.

- template<bool Byteswap = true, typename Type , class = typename std::enable_if<Byteswap>::type>
static unsigned long long int **rbyteswap** (Type *const first, Type *const last)

Swap bytes of a range between pointers.

- template<bool Byteswap = true, typename Type , class = typename std::enable_if<(Byteswap) && (!std::is_pointer<Type>-::value)>::type>
static unsigned long long int **rbyteswap** (const Type &first, const Type &last, typename std::iterator_traits< Type >::iterator_category *=nullptr)

Swap bytes of a range between iterators.

Write stream

- template<bool Byteswap = false>
static bool **write** (std::ostream &stream)

Write nothing to stream.

- template<bool Byteswap = false>
static bool **write** (std::ostream &stream, const std::tuple<> &)

Write an empty tuple to stream.

- template<bool Byteswap = false, typename Type >
static bool **write** (std::ostream &stream, const Type &variable)

Write a single variable to stream.

- template<bool Byteswap = false, typename Type , typename... Types>
static bool **write** (std::ostream &stream, const Type &variable, const Types &...variables)

Write several variables to stream.

- template<bool Byteswap = false, typename Type , std::size_t Size>
static bool **write** (std::ostream &stream, const std::array< Type, Size > &container)

Write an array to stream.

- template<bool Byteswap = false, unsigned int Current = 0, typename... Types>
static bool **write** (std::ostream &stream, const std::tuple< Types...> &container)

Write a tuple to stream.

- template<bool Byteswap = false>
static bool **rwrite** (std::ostream &stream, const std::nullptr_t=nullptr, const std::nullptr_t=nullptr)

Write an empty range to stream.

- template<bool Byteswap = false, typename Type >
static bool **rwrite** (std::ostream &stream, const Type *const first, const Type *const last)

Write a range between pointers to stream.

- template<bool Byteswap = false, typename Type , class = typename std::enable_if<!std::is_pointer<Type>::value>::type>
static bool **rwrite** (std::ostream &stream, const Type &first, const Type &last, typename std::iterator_traits< Type >::iterator_category *=nullptr)

Write a range between iterators to stream.

Read stream

- template<bool Byteswap = false>
static bool **read** (std::istream &stream)
Read nothing from stream.
- template<bool Byteswap = false>
static bool **read** (std::istream &stream, std::tuple<> &)
Read an empty tuple from stream.
- template<bool Byteswap = false, typename Type >
static bool **read** (std::istream &stream, Type &variable)
Read a single variable from stream.
- template<bool Byteswap = false, typename Type , typename... Types>
static bool **read** (std::istream &stream, Type &variable, Types &...variables)
Read several variables from stream.
- template<bool Byteswap = false, typename Type , std::size_t Size>
static bool **read** (std::istream &stream, std::array< Type, Size > &container)
Read an array from stream.
- template<bool Byteswap = false, unsigned int Current = 0, typename... Types>
static bool **read** (std::istream &stream, std::tuple< Types...> &container)
Read a tuple from stream.
- template<bool Byteswap = false>
static bool **rread** (std::istream &stream, Type *const first, Type *const last)
Read a range between pointers from stream.
- template<bool Byteswap = false, typename Type , class = typename std::enable_if<!std::is_pointer<Type>::value>::type>
static bool **read** (std::istream &stream, const Type &first, const Type &last, typename std::iterator_traits< Type >::iterator_category *=nullptr)
Read a range between iterators from stream.

Write buffer

- template<bool Byteswap = false>
static char *& **write** (char *&buffer)
Write nothing to buffer.
- template<bool Byteswap = false>
static char *& **write** (char *&buffer, const std::tuple<> &)
Write an empty tuple to buffer.
- template<bool Byteswap = false, typename Type >
static char *& **write** (char *&buffer, const Type &variable)
Write a single variable to buffer.
- template<bool Byteswap = false, typename Type , typename... Types>
static char *& **write** (char *&buffer, const Type &variable, const Types &...variables)
Write several variables to buffer.
- template<bool Byteswap = false, typename Type , std::size_t Size>
static char *& **write** (char *&buffer, const std::array< Type, Size > &container)
Write an array to buffer.
- template<bool Byteswap = false, unsigned int Current = 0, typename... Types>
static char *& **write** (char *&buffer, const std::tuple< Types...> &container)
Write a tuple to buffer.
- template<bool Byteswap = false>
static char *& **rwrite** (char *&buffer, const std::nullptr_t=nullptr, const std::nullptr_t=nullptr)
Write an empty range to buffer.
- template<bool Byteswap = false, typename Type >
static char *& **rwrite** (char *&buffer, const Type *const first, const Type *const last)
Write a range between pointers to buffer.
- template<bool Byteswap = false, typename Type , class = typename std::enable_if<!std::is_pointer<Type>::value>::type>
static char *& **rwrite** (char *&buffer, const Type &first, const Type &last, typename std::iterator_traits< Type >::iterator_category *=nullptr)

Write a range between iterators to buffer.

Read buffer

- template<bool Byteswap = false>
static char *& **read** (char *&buffer)
Read nothing from buffer.
- template<bool Byteswap = false>
static char *& **read** (char *&buffer, std::tuple<> &)
Read an empty tuple from buffer.
- template<bool Byteswap = false, typename Type >
static char *& **read** (char *&buffer, Type &variable)
Read a single variable from buffer.
- template<bool Byteswap = false, typename Type , typename... Types>
static char *& **read** (char *&buffer, Type &variable, Types &...variables)
Read several variables from buffer.
- template<bool Byteswap = false, typename Type , std::size_t Size>
static char *& **read** (char *&buffer, std::array< Type, Size > &container)
Read an array from buffer.
- template<bool Byteswap = false, unsigned int Current = 0, typename... Types>
static char *& **read** (char *&buffer, std::tuple< Types...> &container)
Read a tuple from buffer.
- template<bool Byteswap = false>
static char *& **read** (char *&buffer, const std::nullptr_t=nullptr, const std::nullptr_t=nullptr)
Read an empty range from buffer.
- template<bool Byteswap = false, typename Type >
static char *& **read** (char *&buffer, Type *const first, Type *const last)
Read a range between pointers from buffer.
- template<bool Byteswap = false, typename Type , class = typename std::enable_if<std::is_pointer<Type>::value>::type>
static char *& **read** (char *&buffer, const Type &first, const Type &last, typename std::iterator_traits< Type >::iterator_category *=nullptr)
Read a range between iterators from buffer.

Nullification

- static unsigned int **nullify** ()
Nullify nothing.
- static unsigned int **nullify** (std::tuple<> &)
Nullify an empty tuple.
- template<typename Type >
static unsigned int **nullify** (Type &variable)
Nullify a single variable.
- template<typename Type , typename... Types>
static unsigned int **nullify** (Type &variable, Types &...variables)
Nullify several variables.
- template<typename Type , std::size_t Size>
static unsigned int **nullify** (std::array< Type, Size > &container)
Nullify an array.
- template<unsigned int Current = 0, typename... Types>
static unsigned int **nullify** (std::tuple< Types...> &container)
Nullify a tuple.
- static unsigned long long int **rnullify** (const std::nullptr_t=nullptr, const std::nullptr_t=nullptr)
Nullify an empty range.
- template<typename Type >
static unsigned long long int **rnullify** (Type *const first, Type *const last)
Nullify a range between pointers.
- template<typename Type , class = typename std::enable_if<std::is_pointer<Type>::value>::type>
static unsigned long long int **rnullify** (const Type &first, const Type &last, typename std::iterator_traits< Type >::iterator_category *=nullptr)

Nullify a range between iterators.

Equalization

- template<typename Reference >
static unsigned int **equalize** (const Reference &reference)
Equalize nothing.
- template<typename Reference >
static unsigned int **equalize** (const Reference &reference, std::tuple<> &)
Equalize an empty tuple.
- template<typename Reference , typename Type >
static unsigned int **equalize** (const Reference &reference, Type &variable)
Equalize a single variable.
- template<typename Reference , typename Type , typename... Types>
static unsigned int **equalize** (const Reference &reference, Type &variable, Types &...variables)
Equalize several variables.
- template<typename Reference , typename Type , std::size_t Size>
static unsigned int **equalize** (const Reference &reference, std::array< Type, Size > &container)
Equalize an array.
- template<unsigned int Current = 0, typename Reference , typename... Types>
static unsigned int **equalize** (const Reference &reference, std::tuple< Types...> &container)
Equalize a tuple.
- template<typename Reference >
static unsigned long long int **requalize** (const Reference &reference, const std::nullptr_t=nullptr, const std::nullptr_t=nullptr)
Equalize an empty range.
- template<typename Reference , typename Type >
static unsigned long long int **requalize** (const Reference &reference, Type *const first, Type *const last)
Equalize a range between pointers.
- template<typename Reference , typename Type , class = typename std::enable_if<!std::is_pointer<Type>::value>::type>
static unsigned long long int **requalize** (const Reference &reference, const Type &first, const Type &last, typename std::iterator_traits< Type >::iterator_category *=nullptr)
Equalize a range between iterators.

Hexification

- template<bool Byteswap = false, bool Uppercase = false>
static std::string **hexify** ()
Convert nothing to a hexadecimal string.
- template<bool Byteswap = false, bool Uppercase = false>
static std::string **hexify** (const std::tuple<> &)
Convert an empty tuple to a hexadecimal string.
- template<bool Byteswap = false, bool Uppercase = false, typename Type >
static std::string **hexify** (const Type &variable)
Convert a single variable to a hexadecimal string.
- template<bool Byteswap = false, bool Uppercase = false, typename Type , typename... Types>
static std::string **hexify** (const Type &variable, const Types &...variables)
Convert several variables to a hexadecimal string.
- template<bool Byteswap = false, bool Uppercase = false, typename Type , std::size_t Size>
static std::string **hexify** (const std::array< Type, Size > &container)
Convert an array to a hexadecimal string.
- template<bool Byteswap = false, bool Uppercase = false, unsigned int Current = 0, typename... Types>
static std::string **hexify** (const std::tuple< Types...> &container)
Convert a tuple to a hexadecimal string.
- template<bool Byteswap = false, bool Uppercase = false>
static std::string **rhexify** (const std::nullptr_t=nullptr, const std::nullptr_t=nullptr, const std::string &separator=" ")
Convert an empty range to a hexadecimal string.

- template<bool Byteswap = false, bool Uppercase = false, typename Type >
 static std::string **rhexify** (const Type *const first, const Type *const last, const std::string &separator=" ")
Convert a range between pointers to a hexadecimal string.
- template<bool Byteswap = false, bool Uppercase = false, typename Type , class = typename std::enable_if<!std::is_pointer<Type>-::value>::type>
 static std::string **rhexify** (const Type &first, const Type &last, const std::string &separator=" ", typename std::iterator_traits< Type >::iterator_category *=nullptr)
Convert a range between iterators to a hexadecimal string.

Stringification

- template<unsigned int Base = 10, char Leading = char()>
 static std::string **stringify** ()
Convert nothing to a string.
- template<unsigned int Base = 10, char Leading = char()>
 static std::string **stringify** (const std::tuple<> &)
Convert an empty tuple to a string.
- template<unsigned int Base = 10, char Leading = char(), typename Type >
 static std::string **stringify** (const Type &variable)
Convert a single variable to a string.
- template<unsigned int Base = 10, char Leading = char(), typename Type , typename... Types>
 static std::string **stringify** (const Type &variable, const Types &...variables)
Convert several variables to a string.
- template<unsigned int Base = 10, char Leading = char(), typename Type , std::size_t Size>
 static std::string **stringify** (const std::array< Type, Size > &container)
Convert an array to a string.
- template<unsigned int Base = 10, char Leading = char(), unsigned int Current = 0, typename... Types>
 static std::string **stringify** (const std::tuple< Types...> &container)
Convert a tuple to a string.
- template<unsigned int Base = 10, char Leading = char()>
 static std::string **rstringify** (const std::nullptr_t=nullptr, const std::nullptr_t=nullptr, const std::string &separator=" ")
Convert an empty range to a string.
- template<unsigned int Base = 10, char Leading = char(), typename Type >
 static std::string **rstringify** (const Type *const first, const Type *const last, const std::string &separator=" ")
Convert a range between pointers to a string.
- template<unsigned int Base = 10, char Leading = char(), typename Type , class = typename std::enable_if<!std::is_pointer<Type>-::value>::type>
 static std::string **rstringify** (const Type &first, const Type &last, const std::string &separator=" ", typename std::iterator_traits< Type >::iterator_category *=nullptr)
Convert a range between iterators to a string.

Print

- static bool **print** (std::ostream &stream)
Print nothing to stream.
- static bool **print** (std::ostream &stream, const std::tuple<> &)
Print an empty tuple to stream.
- template<typename Type >
 static bool **print** (std::ostream &stream, const Type &variable)
Print a single variable to stream.
- template<typename Type , typename... Types>
 static bool **print** (std::ostream &stream, const Type &variable, const Types &...variables)
Print several variables to stream.
- template<typename Type , std::size_t Size>
 static bool **print** (std::ostream &stream, const std::array< Type, Size > &container)
Print an array to stream.

- template<unsigned int Current = 0, typename... Types>
static bool **print** (std::ostream &stream, const std::tuple< Types...> &container)
Print a tuple to stream.
- static bool **rprint** (std::ostream &stream, const std::nullptr_t=nullptr, const std::nullptr_t=nullptr)
Print an empty range to stream.
- template<typename Type >
static bool **rprint** (std::ostream &stream, const Type *const first, const Type *const last)
Print a range between pointers to stream.
- template<typename Type , class = typename std::enable_if<!std::is_pointer<Type>::value>::type>
static bool **rprint** (std::ostream &stream, const Type &first, const Type &last, typename std::iterator_traits< Type >::iterator_category *=nullptr)
Print a range between iterators to stream.

Scan

- static bool **scan** (std::istream &stream)
Scan nothing from stream.
- static bool **scan** (std::istream &stream, std::tuple<> &)
Scan an empty tuple from stream.
- template<typename Type >
static bool **scan** (std::istream &stream, Type &variable)
Scan a single variable from stream.
- template<typename Type , typename... Types>
static bool **scan** (std::istream &stream, Type &variable, Types &...variables)
Scan several variables from stream.
- template<typename Type , std::size_t Size>
static bool **scan** (std::istream &stream, std::array< Type, Size > &container)
Scan an array from stream.
- template<unsigned int Current = 0, typename... Types>
static bool **scan** (std::istream &stream, std::tuple< Types...> &container)
Scan a tuple from stream.
- static bool **rscan** (std::istream &stream, const std::nullptr_t=nullptr, const std::nullptr_t=nullptr)
Scan an empty range from stream.
- template<typename Type >
static bool **rscan** (std::istream &stream, Type *const first, Type *const last)
Scan a range between pointers from stream.
- template<typename Type , class = typename std::enable_if<!std::is_pointer<Type>::value>::type>
static bool **rscan** (std::istream &stream, const Type &first, const Type &last, typename std::iterator_traits< Type >::iterator_category *=nullptr)
Scan a range between iterators from stream.

Test

- static int **example** ()
Example function.

6.19.1 Detailed Description

Set of basic operations on binary data related to IO.

Provides a wide range of overloaded utilities to handle and format data for reading and writing tasks : byteswap, size, extraction from tuples...

6.19.2 Member Function Documentation

6.19.2.1 template<typename Type > constexpr bool magrathea::DataHandler::array (const Type &) [static]

Is not an array.

Returns false as the passed argument is not a standard array.

Returns

Whether the argument is a standard array.

6.19.2.2 template<typename Type , std::size_t Size> constexpr bool magrathea::DataHandler::array (const std::array< Type, Size > &) [static]

Is an array.

Returns true as the passed argument is a standard array.

Returns

Whether the argument is a standard array.

6.19.2.3 template<typename Type > constexpr unsigned int magrathea::DataHandler::bytesize (const std::nullptr_t = nullptr) [static]

Size in bytes of a single type.

Computes the size in bytes of the passed type. The returned size is either the result of `sizeof()` or 0 if the type is void. Note that if the passed type is a templated type, like a tuple, the inner arguments are not extracted, and the result is simply the same as `sizeof()`.

Returns

Size in bytes.

6.19.2.4 template<typename Type , typename... Types, class > constexpr unsigned int magrathea::DataHandler::bytesize () [static]

Size in bytes of a single type.

Computes the size in bytes of the passed types. The returned size is the sum of the sizes of each type.

Returns

Size in bytes.

6.19.2.5 constexpr unsigned int magrathea::DataHandler::bytesize (const std::tuple<> &) [static]

Size in bytes of an empty tuple.

Function overload that does nothing.

Returns

Size in bytes.

6.19.2.6 template<typename Type > constexpr unsigned int magrathea::DataHandler::bytesize (const Type & variable) [static]

Size in bytes of a single variable.

Computes the size in bytes of the passed variable.

Template Parameters

Type	(Variable type.)
------	------------------

Parameters

in	variable	Variable.
----	----------	-----------

Returns

Size in bytes.

6.19.2.7 template<typename Type , typename... Types> constexpr unsigned int magrathea::DataHandler::bytesize (const Type & variable, const Types &... variables) [static]

Size in bytes of several variables.

Computes the sum of the size in bytes of the passed variables.

Template Parameters

Type	(Variable type.)
Types	(Variadic types.)

Parameters

in	variable	Variable.
in	variables	Variables.

Returns

Size in bytes.

6.19.2.8 template<typename Type , std::size_t Size> constexpr unsigned int magrathea::DataHandler::bytesize (const std::array< Type, Size > & container) [static]

Size in bytes of an array.

Computes the sum of the size in bytes of elements of the passed array.

Template Parameters

Type	(Array type.)
Size	(Array size.)

Parameters

in	container	Array.
----	-----------	--------

Returns

Size in bytes.

6.19.2.9 `template<unsigned int Current, typename... Types> constexpr unsigned int magrathea::DataHandler::bytesize (const std::tuple<Types...> & container) [static]`

Size in bytes of a tuple.

Computes the sum of the size in bytes of elements of the passed tuple.

Template Parameters

<i>Current</i>	(Current level of recursion.)
<i>Types</i>	(Variadic types.)

Parameters

<i>in</i>	<i>container</i>	Tuple.
-----------	------------------	--------

Returns

Size in bytes.

6.19.2.10 `template<bool Byteswap, typename... Types, class > unsigned int magrathea::DataHandler::byteswap (const Types & ...) [inline], [static]`

Do not swap bytes.

Function overload that does nothing.

Template Parameters

<i>Byteswap</i>	Do not swap endianness if false.
<i>Types</i>	(Variadic types.)

Returns

Number of elements successfully byteswapped.

6.19.2.11 `template<bool Byteswap, class > unsigned int magrathea::DataHandler::byteswap () [inline], [static]`

Swap bytes of nothing.

Function overload that does nothing.

Template Parameters

<i>Byteswap</i>	Swap endianness if true.
-----------------	--------------------------

Returns

Number of elements successfully byteswapped.

6.19.2.12 `template<bool Byteswap, class > unsigned int magrathea::DataHandler::byteswap (std::tuple<> &) [inline], [static]`

Swap bytes of an empty tuple.

Function overload that does nothing.

Template Parameters

<i>Byteswap</i>	Swap endianness if true.
-----------------	--------------------------

Returns

Number of elements successfully byteswapped.

6.19.2.13 `template<bool Byteswap, typename Type, class > unsigned int magrathea::DataHandler::byteswap (Type & variable) [inline], [static]`

Swap bytes of a single variable.

Inverts the order of bytes of the passed variable to change the endianness.

Template Parameters

<i>Byteswap</i>	Swap endianness if true.
<i>Type</i>	(Variable type.)

Parameters

<i>in, out</i>	<i>variable</i>	Variable.
----------------	-----------------	-----------

Returns

Number of elements successfully byteswapped.

6.19.2.14 `template<bool Byteswap, typename Type, typename... Types, class > unsigned int magrathea::DataHandler::byteswap (Type & variable, Types &... variables) [inline], [static]`

Swap bytes of several variables.

Inverts the order of bytes of the passed variables to change the endianness.

Template Parameters

<i>Byteswap</i>	Swap endianness if true.
<i>Type</i>	(Variable type.)
<i>Types</i>	(Variadic types.)

Parameters

<i>in, out</i>	<i>variable</i>	Variable.
<i>in, out</i>	<i>variables</i>	Variables.

Returns

Number of elements successfully byteswapped.

6.19.2.15 `template<bool Byteswap, typename Type, std::size_t Size, class > unsigned int magrathea::DataHandler::byteswap (std::array< Type, Size > & container) [inline], [static]`

Swap bytes of an array.

Inverts the order of bytes of each element of the passed array to change the endianness.

Template Parameters

<i>Byteswap</i>	Swap endianness if true.
<i>Type</i>	(Array type.)
<i>Size</i>	(Array size.)

Parameters

<code>in, out</code>	<code>container</code>	Array.
----------------------	------------------------	--------

Returns

Number of elements successfully byteswapped.

6.19.2.16 `template<bool Byteswap, unsigned int Current, typename... Types, class > unsigned int magrathea::DataHandler::byteswap (std::tuple< Types...> & container) [inline], [static]`

Swap bytes of a tuple.

Inverts the order of bytes of each element of the passed tuple to change the endianness.

Template Parameters

<i>Byteswap</i>	Swap endianness if true.
<i>Current</i>	(Current level of recursion.)
<i>Types</i>	(Variadic types.)

Parameters

<code>in, out</code>	<code>container</code>	Tuple.
----------------------	------------------------	--------

Returns

Number of elements successfully byteswapped.

6.19.2.17 `template<typename Reference > unsigned int magrathea::DataHandler::equalize (const Reference & reference) [inline], [static]`

Equalize nothing.

Function overload that does nothing.

Template Parameters

<i>Reference</i>	(Reference type.)
------------------	-------------------

Parameters

in	<i>reference</i>	Reference value.
----	------------------	------------------

Returns

Number of elements successfully set to the reference value.

6.19.2.18 template<typename Reference > unsigned int magrathea::DataHandler::equalize (const Reference & *reference*, std::tuple<> &) [inline], [static]

Equalize an empty tuple.

Function overload that does nothing.

Template Parameters

<i>Reference</i>	(Reference type.)
------------------	-------------------

Parameters

in	<i>reference</i>	Reference value.
----	------------------	------------------

Returns

Number of elements successfully set to the reference value.

6.19.2.19 template<typename Reference , typename Type > unsigned int magrathea::DataHandler::equalize (const Reference & *reference*, Type & *variable*) [inline], [static]

Equalize a single variable.

Set the passed variable to the reference value.

Template Parameters

<i>Reference</i>	(Reference type.)
<i>Type</i>	(Variable type.)

Parameters

in	<i>reference</i>	Reference value.
out	<i>variable</i>	Variable.

Returns

Number of elements successfully set to the reference value.

6.19.2.20 template<typename Reference , typename Type , typename... Types> unsigned int magrathea::DataHandler::equalize (const Reference & *reference*, Type & *variable*, Types &... *variables*) [inline], [static]

Equalize several variables.

Set the passed variables to the reference value.

Template Parameters

<i>Reference</i>	(Reference type.)
<i>Type</i>	(Variable type.)
<i>Types</i>	(Variadic types.)

Parameters

<i>in</i>	<i>reference</i>	Reference value.
<i>out</i>	<i>variable</i>	Variable.
<i>out</i>	<i>variables</i>	Variables.

Returns

Number of elements successfully set to the reference value.

```
6.19.2.21 template<typename Reference , typename Type , std::size_t Size> unsigned int magrathea::DataHandler::equalize (
    const Reference & reference, std::array< Type, Size > & container ) [inline], [static]
```

Equalize an array.

Set each element of the passed array to the reference value.

Template Parameters

<i>Reference</i>	(Reference type.)
<i>Type</i>	(Array type.)
<i>Size</i>	(Array size.)

Parameters

<i>in</i>	<i>reference</i>	Reference value.
<i>out</i>	<i>container</i>	Array.

Returns

Number of elements successfully set to the reference value.

```
6.19.2.22 template<unsigned int Current, typename Reference , typename... Types> unsigned int
magrathea::DataHandler::equalize ( const Reference & reference, std::tuple< Types...> & container )
[inline], [static]
```

Equalize a tuple.

Set each element of the passed tuple to the reference value.

Template Parameters

<i>Current</i>	(Current level of recursion.)
<i>Reference</i>	(Reference type.)
<i>Types</i>	(Variadic types.)

Parameters

<i>in</i>	<i>reference</i>	Reference value.
<i>out</i>	<i>container</i>	Tuple.

Returns

Number of elements successfully set to the reference value.

6.19.2.23 int magrathea::DataHandler::example() [static]

Example function.

Tests and demonstrates the use of [Timer](#).

Returns

0 if no error.

6.19.2.24 template<bool Byteswap, bool Uppercase> std::string magrathea::DataHandler::hexify() [static]

Convert nothing to a hexadecimal string.

Function overload that does nothing.

Template Parameters

<i>Byteswap</i>	Swap bytes of hexadecimal conversion if true.
<i>Uppercase</i>	Force uppercase letters for hexadecimal representation if true.

Returns

Hexadecimal representation of the input.

6.19.2.25 template<bool Byteswap, bool Uppercase> std::string magrathea::DataHandler::hexify(const std::tuple<> &) [static]

Convert an empty tuple to a hexadecimal string.

Function overload that does nothing.

Template Parameters

<i>Byteswap</i>	Swap bytes of hexadecimal conversion if true.
<i>Uppercase</i>	Force uppercase letters for hexadecimal representation if true.

Returns

Hexadecimal representation of the input.

6.19.2.26 template<bool Byteswap, bool Uppercase, typename Type > std::string magrathea::DataHandler::hexify(const Type & variable) [static]

Convert a single variable to a hexadecimal string.

Converts bytes of the passed variable to their hexadecimal representation and returns the associated string.

Template Parameters

<i>Byteswap</i>	Swap bytes of hexadecimal conversion if true.
<i>Uppercase</i>	Force uppercase letters for hexadecimal representation if true.
<i>Type</i>	(Variable type.)

Parameters

<i>in</i>	<i>variable</i>	Variable.
-----------	-----------------	-----------

Returns

Hexadecimal representation of the input.

6.19.2.27 template<bool Byteswap, bool Uppercase, typename Type , typename... Types> std::string magrathea::DataHandler::hexify (const Type & *variable*, const Types &... *variables*) [static]

Convert several variables to a hexadecimal string.

Converts bytes of the passed variables to their hexadecimal representation and returns the associated string.

Template Parameters

<i>Byteswap</i>	Swap bytes of hexadecimal conversion if true.
<i>Uppercase</i>	Force uppercase letters for hexadecimal representation if true.
<i>Type</i>	(Variable type.)
<i>Types</i>	(Variadic types.)

Parameters

<i>in</i>	<i>variable</i>	Variable.
<i>in</i>	<i>variables</i>	Variables.

Returns

Hexadecimal representation of the input.

6.19.2.28 template<bool Byteswap, bool Uppercase, typename Type , std::size_t Size> std::string magrathea::DataHandler::hexify (const std::array< Type, Size > & *container*) [static]

Convert an array to a hexadecimal string.

Converts bytes of each element of the passed array to their hexadecimal representation and returns the associated string.

Template Parameters

<i>Byteswap</i>	Swap bytes of hexadecimal conversion if true.
<i>Uppercase</i>	Force uppercase letters for hexadecimal representation if true.
<i>Type</i>	(Array type.)
<i>Size</i>	(Array size.)

Parameters

<i>in</i>	<i>container</i>	Array.
-----------	------------------	--------

Returns

Hexadecimal representation of the input.

6.19.2.29 `template<bool Byteswap, bool Uppercase, unsigned int Current, typename... Types> std::string magrathea::DataHandler::hexify (const std::tuple< Types...> & container) [static]`

Convert a tuple to a hexadecimal string.

Converts bytes of each element of the passed tuple to their hexadecimal representation and returns the associated string.

Template Parameters

<i>Byteswap</i>	Swap bytes of hexadecimal conversion if true.
<i>Uppercase</i>	Force uppercase letters for hexadecimal representation if true.
<i>Current</i>	(Current level of recursion.)
<i>Types</i>	(Variadic types.)

Parameters

<code>in</code>	<code>container</code>	Tuple.
-----------------	------------------------	--------

Returns

Hexadecimal representation of the input.

6.19.2.30 `unsigned int magrathea::DataHandler::nullify () [inline], [static]`

Nullify nothing.

Function overload that does nothing.

Returns

Number of elements successfully set to zero.

6.19.2.31 `unsigned int magrathea::DataHandler::nullify (std::tuple<> &) [inline], [static]`

Nullify an empty tuple.

Function overload that does nothing.

Returns

Number of elements successfully set to zero.

6.19.2.32 `template<typename Type > unsigned int magrathea::DataHandler::nullify (Type & variable) [inline], [static]`

Nullify a single variable.

Calls the constructor of the passed variable to nullify it.

Template Parameters

<i>Type</i>	(Variable type.)
-------------	------------------

Parameters

<code>out</code>	<code>variable</code>	Variable.
------------------	-----------------------	-----------

Returns

Number of elements successfully set to zero.

**6.19.2.33 template<typename Type , typename... Types> unsigned int magrathea::DataHandler::nullify (`Type & variable,`
`Types &... variables`) [inline], [static]**

Nullify several variables.

Calls the constructor of the passed variables to nullify them.

Template Parameters

<code>Type</code>	(Variable type.)
<code>Types</code>	(Variadic types.)

Parameters

<code>out</code>	<code>variable</code>	Variable.
<code>out</code>	<code>variables</code>	Variables.

Returns

Number of elements successfully set to zero.

**6.19.2.34 template<typename Type , std::size_t Size> unsigned int magrathea::DataHandler::nullify (`std::array< Type, Size >`
`& container`) [inline], [static]**

Nullify an array.

Calls the constructor of each element of the passed array to nullify it.

Template Parameters

<code>Type</code>	(Array type.)
<code>Size</code>	(Array size.)

Parameters

<code>out</code>	<code>container</code>	Array.
------------------	------------------------	--------

Returns

Number of elements successfully set to zero.

**6.19.2.35 template<unsigned int Current, typename... Types> unsigned int magrathea::DataHandler::nullify (`std::tuple<`
`Types...> & container`) [inline], [static]**

Nullify a tuple.

Calls the constructor of each element of the passed tuple to nullify it.

Template Parameters

<i>Current</i>	(Current level of recursion.)
<i>Types</i>	(Variadic types.)

Parameters

<i>out</i>	<i>container</i>	Tuple.
------------	------------------	--------

Returns

Number of elements successfully set to zero.

6.19.2.36 bool magrathea::DataHandler::print (std::ostream & stream) [inline], [static]

Print nothing to stream.

Function overload that does nothing.

Parameters

<i>in,out</i>	<i>stream</i>	Output stream.
---------------	---------------	----------------

Returns

True if the stream has no failure, false otherwise.

6.19.2.37 bool magrathea::DataHandler::print (std::ostream & stream, const std::tuple<>&) [inline], [static]

Print an empty tuple to stream.

Function overload that does nothing.

Parameters

<i>in,out</i>	<i>stream</i>	Output stream.
---------------	---------------	----------------

Returns

True if the stream has no failure, false otherwise.

6.19.2.38 template<typename Type > bool magrathea::DataHandler::print (std::ostream & stream, const Type & variable) [inline], [static]

Print a single variable to stream.

Prints the next variable to the stream.

Template Parameters

<i>Type</i>	(Variable type.)
-------------	------------------

Parameters

<i>in,out</i>	<i>stream</i>	Output stream.
<i>in</i>	<i>variable</i>	Variable.

Returns

True if the stream has no failure, false otherwise.

6.19.2.39 template<typename Type , typename... Types> bool magrathea::DataHandler::print (std::ostream & stream, const Type & variable, const Types &... variables) [inline], [static]

Print several variables to stream.

Prints the next variables to the stream.

Template Parameters

Type	(Variable type.)
Types	(Variadic types.)

Parameters

in,out	stream	Output stream.
in	variable	Variable.
in	variables	Variables.

Returns

True if the stream has no failure, false otherwise.

6.19.2.40 template<typename Type , std::size_t Size> bool magrathea::DataHandler::print (std::ostream & stream, const std::array< Type, Size > & container) [inline], [static]

Print an array to stream.

Prints each element of the passed array to the stream.

Template Parameters

Type	(Array type.)
Size	(Array size.)

Parameters

in,out	stream	Output stream.
in	container	Array.

Returns

True if the stream has no failure, false otherwise.

6.19.2.41 template<unsigned int Current, typename... Types> bool magrathea::DataHandler::print (std::ostream & stream, const std::tuple< Types...> & container) [inline], [static]

Print a tuple to stream.

Prints each element of the passed tuple to the stream.

Template Parameters

<i>Current</i>	(Current level of recursion.)
<i>Types</i>	(Variadic types.)

Parameters

<i>in,out</i>	<i>stream</i>	Output stream.
<i>in</i>	<i>container</i>	Tuple.

Returns

True if the stream has no failure, false otherwise.

6.19.2.42 `unsigned long long int magrathea::DataHandler::rbytesize (const std::nullptr_t = nullptr, const std::nullptr_t = nullptr) [inline], [static]`

Size in bytes of an empty range.

Function overload that does nothing.

Returns

Size in bytes.

6.19.2.43 `template<typename Type > unsigned long long int magrathea::DataHandler::rbytesize (const Type *const first, const Type *const last) [inline], [static]`

Size in bytes of a range between pointers.

Computes the sum of the size in bytes of the elements of the range between the passed pointers.

Template Parameters

<i>Type</i>	(Pointer type.)
-------------	-----------------

Parameters

<i>in</i>	<i>first</i>	Pointer to the beginning.
<i>in</i>	<i>last</i>	Pointer to the end.

Returns

Size in bytes.

6.19.2.44 `template<typename Type , class > unsigned long long int magrathea::DataHandler::rbytesize (const Type & first, const Type & last, typename std::iterator_traits< Type >::iterator_category * = nullptr) [inline], [static]`

Size in bytes of a range between iterators.

Computes the sum of the size in bytes of the range between the passed iterators.

Template Parameters

<i>Type</i>	(Iterator type.)
-------------	------------------

Parameters

in	<i>first</i>	Iterator to the beginning.
in	<i>last</i>	Iterator to the end.

Returns

Size in bytes.

6.19.2.45 template<bool Byteswap, typename... Types, class > unsigned int magrathea::DataHandler::rbyteswap (const Types & ...) [inline], [static]

Do not swap bytes of a range.

Function overload that does nothing.

Template Parameters

<i>Byteswap</i>	Do not swap endianness if false.
<i>Types</i>	(Variadic types.)

Returns

Number of elements successfully byteswapped.

6.19.2.46 template<bool Byteswap, class > unsigned long long int magrathea::DataHandler::rbyteswap (const std::nullptr_t = nullptr, const std::nullptr_t = nullptr) [inline], [static]

Swap bytes of an empty range.

Function overload that does nothing.

Template Parameters

<i>Byteswap</i>	Swap endianness if true.
-----------------	--------------------------

Returns

Number of elements successfully byteswapped.

6.19.2.47 template<bool Byteswap, typename Type, class > unsigned long long int magrathea::DataHandler::rbyteswap (Type *const *first*, Type *const *last*) [inline], [static]

Swap bytes of a range between pointers.

Inverts the order of bytes of each element of the range between the passed pointers.

Template Parameters

<i>Byteswap</i>	Swap endianness if true.
<i>Type</i>	(Pointer type.)

Parameters

in	<i>first</i>	Pointer to the beginning.
in	<i>last</i>	Pointer to the end.

Returns

Number of elements successfully byteswapped.

```
6.19.2.48 template<bool Byteswap, typename Type, class > unsigned long long int magrathea::DataHandler::rbyteswap (
    const Type & first, const Type & last, typename std::iterator_traits< Type >::iterator_category * = nullptr )
    [inline], [static]
```

Swap bytes of a range between iterators.

Inverts the order of bytes of each element of the range between the passed iterators.

Template Parameters

<i>Byteswap</i>	Swap endianness if true.
<i>Type</i>	(Iterator type.)

Parameters

<i>in</i>	<i>first</i>	Iterator to the beginning.
<i>in</i>	<i>last</i>	Iterator to the end.

Returns

Number of elements successfully byteswapped.

```
6.19.2.49 template<bool Byteswap> bool magrathea::DataHandler::read ( std::istream & stream ) [inline],
    [static]
```

Read nothing from stream.

Function overload that does nothing.

Template Parameters

<i>Byteswap</i>	Swap endianness after reading if true.
-----------------	--

Parameters

<i>in,out</i>	<i>stream</i>	Input stream.
---------------	---------------	---------------

Returns

True if the stream has no failure, false otherwise.

```
6.19.2.50 template<bool Byteswap> bool magrathea::DataHandler::read ( std::istream & stream, std::tuple<> & )
    [inline], [static]
```

Read an empty tuple from stream.

Function overload that does nothing.

Template Parameters

<i>Byteswap</i>	Swap endianness after reading if true.
-----------------	--

Parameters

<code>in, out</code>	<code>stream</code>	Input stream.
----------------------	---------------------	----------------------

Returns

True if the stream has no failure, false otherwise.

6.19.2.51 template<bool Byteswap, typename Type > bool magrathea::DataHandler::read (std::istream & *stream*, Type & *variable*) [inline], [static]

Read a single variable from stream.

Reads the next variable from the stream and swap bytes if necessary.

Template Parameters

<code>Byteswap</code>	Swap endianness after reading if true.
<code>Type</code>	(Variable type.)

Parameters

<code>in, out</code>	<code>stream</code>	Input stream.
<code>out</code>	<code>variable</code>	Variable.

Returns

True if the stream has no failure, false otherwise.

6.19.2.52 template<bool Byteswap, typename Type , typename... Types> bool magrathea::DataHandler::read (std::istream & *stream*, Type & *variable*, Types &... *variables*) [inline], [static]

Read several variables from stream.

Reads the next variables from the stream and swap bytes if necessary.

Template Parameters

<code>Byteswap</code>	Swap endianness after reading if true.
<code>Type</code>	(Variable type.)
<code>Types</code>	(Variadic types.)

Parameters

<code>in, out</code>	<code>stream</code>	Input stream.
<code>out</code>	<code>variable</code>	Variable.
<code>out</code>	<code>variables</code>	Variables.

Returns

True if the stream has no failure, false otherwise.

6.19.2.53 template<bool Byteswap, typename Type , std::size_t Size> bool magrathea::DataHandler::read (std::istream & *stream*, std::array< Type, Size > & *container*) [inline], [static]

Read an array from stream.

Reads each element of the passed array from the stream and swap bytes if necessary.

Template Parameters

<i>Byteswap</i>	Swap endianness after reading if true.
<i>Type</i>	(Array type.)
<i>Size</i>	(Array size.)

Parameters

<i>in,out</i>	<i>stream</i>	Input stream.
<i>out</i>	<i>container</i>	Array.

Returns

True if the stream has no failure, false otherwise.

6.19.2.54 `template<bool Byteswap, unsigned int Current, typename... Types> bool magrathea::DataHandler::read (std::istream & stream, std::tuple< Types... > & container) [inline], [static]`

Read a tuple from stream.

Reads each element of the passed tuple from the stream and swap bytes if necessary.

Template Parameters

<i>Byteswap</i>	Swap endianness after reading if true.
<i>Current</i>	(Current level of recursion.)
<i>Types</i>	(Variadic types.)

Parameters

<i>in,out</i>	<i>stream</i>	Input stream.
<i>out</i>	<i>container</i>	Tuple.

Returns

True if the stream has no failure, false otherwise.

6.19.2.55 `template<bool Byteswap> char *& magrathea::DataHandler::read (char *& buffer) [inline], [static]`

Read nothing from buffer.

Function overload that does nothing.

Template Parameters

<i>Byteswap</i>	Swap endianness after reading if true.
-----------------	--

Parameters

<i>in,out</i>	<i>buffer</i>	Input buffer.
---------------	---------------	-------------------------------

Returns

Pointer to the new position in the buffer.

6.19.2.56 template<bool Byteswap> char *& magrathea::DataHandler::read (char *& *buffer*, std::tuple<> &) [inline], [static]

Read an empty tuple from buffer.

Function overload that does nothing.

Template Parameters

<i>Byteswap</i>	Swap endianness after reading if true.
-----------------	--

Parameters

<i>in,out</i>	<i>buffer</i>	Input buffer.
---------------	---------------	---------------

Returns

Pointer to the new position in the buffer.

6.19.2.57 template<bool Byteswap, typename Type > char *& magrathea::DataHandler::read (char *& *buffer*, Type & *variable*) [inline], [static]

Read a single variable from buffer.

Reads the next variable from the buffer and swap bytes if necessary.

Template Parameters

<i>Byteswap</i>	Swap endianness after reading if true.
<i>Type</i>	(Variable type.)

Parameters

<i>in,out</i>	<i>buffer</i>	Input buffer.
<i>out</i>	<i>variable</i>	Variable.

Returns

Pointer to the new position in the buffer.

6.19.2.58 template<bool Byteswap, typename Type , typename... Types> char *& magrathea::DataHandler::read (char *& *buffer*, Type & *variable*, Types &... *variables*) [inline], [static]

Read several variables from buffer.

Reads the next variables from the buffer and swap bytes if necessary.

Template Parameters

<i>Byteswap</i>	Swap endianness after reading if true.
<i>Type</i>	(Variable type.)
<i>Types</i>	(Variadic types.)

Parameters

<i>in, out</i>	<i>buffer</i>	Input buffer.
<i>out</i>	<i>variable</i>	Variable.
<i>out</i>	<i>variables</i>	Variables.

Returns

Pointer to the new position in the buffer.

6.19.2.59 `template<bool Byteswap, typename Type , std::size_t Size> char *& magrathea::DataHandler::read (char *& buffer, std::array< Type, Size > & container) [inline], [static]`

Read an array from buffer.

Reads each element of the passed array from the buffer and swap bytes if necessary.

Template Parameters

<i>Byteswap</i>	Swap endianness after reading if true.
<i>Type</i>	(Array type.)
<i>Size</i>	(Array size.)

Parameters

<i>in, out</i>	<i>buffer</i>	Input buffer.
<i>out</i>	<i>container</i>	Array.

Returns

Pointer to the new position in the buffer.

6.19.2.60 `template<bool Byteswap, unsigned int Current, typename... Types> char *& magrathea::DataHandler::read (char *& buffer, std::tuple< Types... > & container) [inline], [static]`

Read a tuple from buffer.

Reads each element of the passed tuple from the buffer and swap bytes if necessary.

Template Parameters

<i>Byteswap</i>	Swap endianness after reading if true.
<i>Current</i>	(Current level of recursion.)
<i>Types</i>	(Variadic types.)

Parameters

<i>in, out</i>	<i>buffer</i>	Input buffer.
<i>out</i>	<i>container</i>	Tuple.

Returns

Pointer to the new position in the buffer.

6.19.2.61 `template<typename Reference > unsigned long long int magrathea::DataHandler::equalize (const Reference & reference, const std::nullptr_t =nullptr, const std::nullptr_t =nullptr) [inline], [static]`

Equalize an empty range.

Function overload that does nothing.

Template Parameters

<i>Reference</i>	(Reference type.)
------------------	-------------------

Parameters

in	<i>reference</i>	Reference value.
----	------------------	------------------

Returns

Number of elements successfully set to the reference value.

6.19.2.62 `template<typename Reference , typename Type > unsigned long long int magrathea::DataHandler::equalize (const Reference & reference, Type *const first, Type *const last) [inline], [static]`

Equalize a range between pointers.

Set each element of the range between the passed pointers to the reference value.

Template Parameters

<i>Reference</i>	(Reference type.)
<i>Type</i>	(Pointer type.)

Parameters

in	<i>reference</i>	Reference value.
in	<i>first</i>	Pointer to the beginning.
in	<i>last</i>	Pointer to the end.

Returns

Number of elements successfully set to the reference value.

6.19.2.63 `template<typename Reference , typename Type , class > unsigned long long int magrathea::DataHandler::equalize (const Reference & reference, const Type & first, const Type & last, typename std::iterator_traits< Type >::iterator_category * =nullptr) [inline], [static]`

Equalize a range between iterators.

Set each element of the range between the passed iterators to the reference value.

Template Parameters

<i>Reference</i>	(Reference type.)
<i>Type</i>	(Iterator type.)

Parameters

<i>in</i>	<i>reference</i>	Reference value.
<i>in</i>	<i>first</i>	Iterator to the beginning.
<i>in</i>	<i>last</i>	Iterator to the end.

Returns

Number of elements successfully set to the reference value.

```
6.19.2.64 template<bool Byteswap, bool Uppercase> std::string magrathea::DataHandler::rhexify ( const std::nullptr_t = nullptr, const std::nullptr_t = nullptr, const std::string & separator = " " ) [static]
```

Convert an empty range to a hexadecimal string.

Function overload that does nothing.

Template Parameters

<i>Byteswap</i>	Swap bytes of hexadecimal conversion if true.
<i>Uppercase</i>	Force uppercase letters for hexadecimal representation if true.

Parameters

<i>in</i>	<i>separator</i>	String used as a separator between elements.
-----------	------------------	--

Returns

Hexadecimal representation of the input.

```
6.19.2.65 template<bool Byteswap, bool Uppercase, typename Type > std::string magrathea::DataHandler::rhexify ( const Type *const first, const Type *const last, const std::string & separator = " " ) [static]
```

Convert a range between pointers to a hexadecimal string.

Converts bytes of each element of the range between the passed pointers to their hexadecimal representation and returns the associated string.

Template Parameters

<i>Byteswap</i>	Swap bytes of hexadecimal conversion if true.
<i>Uppercase</i>	Force uppercase letters for hexadecimal representation if true.
<i>Type</i>	(Pointer type.)

Parameters

<i>in</i>	<i>first</i>	Pointer to the beginning.
<i>in</i>	<i>last</i>	Pointer to the end.
<i>in</i>	<i>separator</i>	String used as a separator between elements.

Returns

Hexadecimal representation of the input.

6.19.2.66 template<bool Byteswap, bool Uppercase, typename Type , class > std::string magrathea::DataHandler::rhexify (const Type & *first*, const Type & *last*, const std::string & *separator* = " ", typename std::iterator_traits< Type >::iterator_category * = nullptr) [static]

Convert a range between iterators to a hexadecimal string.

Converts bytes of each element of the range between the passed iterators to their hexadecimal representation and returns the associated string.

Template Parameters

<i>Byteswap</i>	Swap bytes of hexadecimal conversion if true.
<i>Uppercase</i>	Force uppercase letters for hexadecimal representation if true.
<i>Type</i>	(Iterator type.)

Parameters

in	<i>first</i>	Iterator to the beginning.
in	<i>last</i>	Iterator to the end.
in	<i>separator</i>	String used as a separator between elements.

Returns

Hexadecimal representation of the input.

6.19.2.67 unsigned long long int magrathea::DataHandler::rnullify (const std::nullptr_t = nullptr, const std::nullptr_t = nullptr) [inline], [static]

Nullify an empty range.

Function overload that does nothing.

Returns

Number of elements successfully set to zero.

6.19.2.68 template<typename Type > unsigned long long int magrathea::DataHandler::rnullify (Type *const *first*, Type *const *last*) [inline], [static]

Nullify a range between pointers.

Calls the constructor of each element of the range between the passed pointers.

Template Parameters

<i>Type</i>	(Pointer type.)
-------------	-----------------

Parameters

in	<i>first</i>	Pointer to the beginning.
in	<i>last</i>	Pointer to the end.

Returns

Number of elements successfully set to zero.

6.19.2.69 template<typename Type , class > unsigned long long int magrathea::DataHandler::rnullify (const Type & *first*, const Type & *last*, typename std::iterator_traits< Type >::iterator_category * =nullptr) [inline], [static]

Nullify a range between iterators.

Calls the constructor of each element of the range between the passed iterators.

Template Parameters

Type	(Iterator type.)
------	------------------

Parameters

in	<i>first</i>	Iterator to the beginning.
in	<i>last</i>	Iterator to the end.

Returns

Number of elements successfully set to zero.

6.19.2.70 bool magrathea::DataHandler::rprint (std::ostream & *stream*, const std::nullptr_t =nullptr, const std::nullptr_t =nullptr) [inline], [static]

Print an empty range to stream.

Function overload that does nothing.

Parameters

in,out	<i>stream</i>	Output stream.
--------	---------------	----------------

Returns

True if the stream has no failure, false otherwise.

6.19.2.71 template<typename Type > bool magrathea::DataHandler::rprint (std::ostream & *stream*, const Type *const *first*, const Type *const *last*) [inline], [static]

Print a range between pointers to stream.

Prints each element of the range between the passed pointers to the stream.

Template Parameters

Type	(Pointer type.)
------	-----------------

Parameters

in,out	<i>stream</i>	Output stream.
in	<i>first</i>	Pointer to the beginning.
in	<i>last</i>	Pointer to the end.

Returns

True if the stream has no failure, false otherwise.

6.19.2.72 `template<typename Type , class > bool magrathea::DataHandler::rprint (std::ostream & stream, const Type & first, const Type & last, typename std::iterator_traits< Type >::iterator_category * =nullptr) [inline], [static]`

Print a range between iterators to stream.

Prints each element of the range between the passed iterators to the stream.

Template Parameters

<i>Type</i>	(Iterator type.)
-------------	------------------

Parameters

<i>in,out</i>	<i>stream</i>	<i>Output</i> stream.
<i>in</i>	<i>first</i>	Iterator to the beginning.
<i>in</i>	<i>last</i>	Iterator to the end.

Returns

True if the stream has no failure, false otherwise.

6.19.2.73 `template<bool Byteswap> bool magrathea::DataHandler::rread (std::istream & stream, const std::nullptr_t = nullptr, const std::nullptr_t =nullptr) [inline], [static]`

Read an empty range from stream.

Function overload that does nothing.

Template Parameters

<i>Byteswap</i>	Swap endianness after reading if true.
-----------------	--

Parameters

<i>in,out</i>	<i>stream</i>	<i>Input</i> stream.
---------------	---------------	----------------------

Returns

True if the stream has no failure, false otherwise.

6.19.2.74 `template<bool Byteswap, typename Type > bool magrathea::DataHandler::rread (std::istream & stream, Type *const first, Type *const last) [inline], [static]`

Read a range between pointers from stream.

Reads each element of the range between the passed pointers from the stream and swap bytes if necessary.

Template Parameters

<i>Byteswap</i>	Swap endianness after reading if true.
<i>Type</i>	(Pointer type.)

Parameters

<i>in,out</i>	<i>stream</i>	Input stream.
<i>in</i>	<i>first</i>	Pointer to the beginning.
<i>in</i>	<i>last</i>	Pointer to the end.

Returns

True if the stream has no failure, false otherwise.

```
6.19.2.75 template<bool Byteswap, typename Type , class > bool magrathea::DataHandler::rread ( std::istream & stream,  
const Type & first, const Type & last, typename std::iterator_traits< Type >::iterator_category * = nullptr )  
[inline], [static]
```

Read a range between iterators from stream.

Reads each element of the range between the passed iterators from the stream and swap bytes if necessary.

Template Parameters

<i>Byteswap</i>	Swap endianness after reading if true.
<i>Type</i>	(Iterator type.)

Parameters

<i>in,out</i>	<i>stream</i>	Input stream.
<i>in</i>	<i>first</i>	Iterator to the beginning.
<i>in</i>	<i>last</i>	Iterator to the end.

Returns

True if the stream has no failure, false otherwise.

```
6.19.2.76 template<bool Byteswap> char *& magrathea::DataHandler::rread ( char *& buffer, const std::nullptr_t =  
nullptr, const std::nullptr_t =nullptr ) [inline], [static]
```

Read an empty range from buffer.

Function overload that does nothing.

Template Parameters

<i>Byteswap</i>	Swap endianness after reading if true.
-----------------	--

Parameters

<i>in,out</i>	<i>buffer</i>	Input buffer.
---------------	---------------	-------------------------------

Returns

Pointer to the new position in the buffer.

```
6.19.2.77 template<bool Byteswap, typename Type > char *& magrathea::DataHandler::rread ( char *& buffer, Type *const  
first, Type *const last ) [inline], [static]
```

Read a range between pointers from buffer.

Reads each element of the range between the passed pointers from the buffer and swap bytes if necessary.

Template Parameters

<i>Byteswap</i>	Swap endianness after reading if true.
<i>Type</i>	(Pointer type.)

Parameters

<i>in,out</i>	<i>buffer</i>	Input buffer.
<i>in</i>	<i>first</i>	Pointer to the beginning.
<i>in</i>	<i>last</i>	Pointer to the end.

Returns

Pointer to the new position in the buffer.

```
6.19.2.78 template<bool Byteswap, typename Type , class > char *& magrathea::DataHandler::rread ( char *& buffer, const
Type & first, const Type & last, typename std::iterator_traits< Type >::iterator_category * = nullptr )
[inline], [static]
```

Read a range between iterators from buffer.

Reads each element of the range between the passed iterators from the buffer and swap bytes if necessary.

Template Parameters

<i>Byteswap</i>	Swap endianness after reading if true.
<i>Type</i>	(Iterator type.)

Parameters

<i>in,out</i>	<i>buffer</i>	Input buffer.
<i>in</i>	<i>first</i>	Iterator to the beginning.
<i>in</i>	<i>last</i>	Iterator to the end.

Returns

Pointer to the new position in the buffer.

```
6.19.2.79 bool magrathea::DataHandler::rscan ( std::istream & stream, const std::nullptr_t = nullptr, const std::nullptr_t
= nullptr ) [inline], [static]
```

Scan an empty range from stream.

Function overload that does nothing.

Parameters

<i>in,out</i>	<i>stream</i>	Input stream.
---------------	---------------	----------------------

Returns

True if the stream has no failure, false otherwise.

6.19.2.80 `template<typename Type > bool magrathea::DataHandler::rscan (std::istream & stream, Type *const first, Type *const last) [inline], [static]`

Scan a range between pointers from stream.

Scans each element of the range between the passed pointers from the stream.

Template Parameters

Type	(Pointer type.)
------	-----------------

Parameters

in, out	stream	Input stream.
in	first	Pointer to the beginning.
in	last	Pointer to the end.

Returns

True if the stream has no failure, false otherwise.

6.19.2.81 `template<typename Type , class > bool magrathea::DataHandler::rscan (std::istream & stream, const Type & first, const Type & last, typename std::iterator_traits< Type >::iterator_category * =nullptr) [inline], [static]`

Scan a range between iterators from stream.

Scans each element of the range between the passed iterators from the stream.

Template Parameters

Type	(Iterator type.)
------	------------------

Parameters

in, out	stream	Input stream.
in	first	Iterator to the beginning.
in	last	Iterator to the end.

Returns

True if the stream has no failure, false otherwise.

6.19.2.82 `unsigned long long int magrathea::DataHandler::rsize (const std::nullptr_t =nullptr, const std::nullptr_t =nullptr) [inline], [static]`

Number of elements of an empty range.

Function overload that does nothing.

Returns

Number of elements.

6.19.2.83 template<typename Type > unsigned long long int magrathea::DataHandler::rsize (const Type *const *first*, const Type *const *last*) [inline], [static]

Number of elements of a range between pointers.

Computes the number of elements of the range between the passed pointers.

Template Parameters

Type	(Pointer type.)
------	-----------------

Parameters

in	<i>first</i>	Pointer to the beginning.
in	<i>last</i>	Pointer to the end.

Returns

Number of elements.

6.19.2.84 template<typename Type , class > unsigned long long int magrathea::DataHandler::rsize (const Type & *first*, const Type & *last*, typename std::iterator_traits< Type >::iterator_category * = nullptr) [inline], [static]

Number of elements of a range between iterators.

Computes the number of elements of the range between the passed iterators.

Template Parameters

Type	(Iterator type.)
------	------------------

Parameters

in	<i>first</i>	Iterator to the beginning.
in	<i>last</i>	Iterator to the end.

Returns

Number of elements.

6.19.2.85 template<unsigned int Base, char Leading> std::string magrathea::DataHandler::rstringify (const std::nullptr_t = nullptr, const std::nullptr_t = nullptr, const std::string & separator = " ") [static]

Convert an empty range to a string.

Function overload that does nothing.

Template Parameters

<i>Base</i>	Base representation for integers : 2, 8, 10 or 16.
<i>Leading</i>	Leading character for integers : 0, 9-13 or 32-126.

Parameters

in	<i>separator</i>	String used as a separator between elements.
----	------------------	--

Returns

String conversion of arguments.

6.19.2.86 template<unsigned int Base, char Leading, typename Type > std::string magrathea::DataHandler::rstringify (const Type *const *first*, const Type *const *last*, const std::string & *separator* = " ") [static]

Convert a range between pointers to a string.

Converts each element of the range between the passed pointers to a string.

Template Parameters

<i>Base</i>	Base representation for integers : 2, 8, 10 or 16.
<i>Leading</i>	Leading character for integers : 0, 9-13 or 32-126.
<i>Type</i>	(Pointer type.)

Parameters

in	<i>first</i>	Pointer to the beginning.
in	<i>last</i>	Pointer to the end.
in	<i>separator</i>	String used as a separator between elements.

Returns

String conversion of arguments.

6.19.2.87 template<unsigned int Base, char Leading, typename Type , class > std::string magrathea::DataHandler::rstringify (const Type & *first*, const Type & *last*, const std::string & *separator* = " ", typename std::iterator_traits< Type >::iterator_category * = nullptr) [static]

Convert a range between iterators to a string.

Converts each element of the range between the passed iterators to a string.

Template Parameters

<i>Base</i>	Base representation for integers : 2, 8, 10 or 16.
<i>Leading</i>	Leading character for integers : 0, 9-13 or 32-126.
<i>Type</i>	(Iterator type.)

Parameters

in	<i>first</i>	Iterator to the beginning.
in	<i>last</i>	Iterator to the end.
in	<i>separator</i>	String used as a separator between elements.

Returns

String conversion of arguments.

6.19.2.88 `template<bool Byteswap> bool magrathea::DataHandler::rwrite (std::ostream & stream, const std::nullptr_t = nullptr, const std::nullptr_t = nullptr) [inline], [static]`

Write an empty range to stream.

Function overload that does nothing.

Template Parameters

<i>Byteswap</i>	Swap endianness for writing if true.
-----------------	--------------------------------------

Parameters

<i>in, out</i>	<i>stream</i>	Output stream.
----------------	---------------	----------------

Returns

True if the stream has no failure, false otherwise.

6.19.2.89 `template<bool Byteswap, typename Type > bool magrathea::DataHandler::rwrite (std::ostream & stream, const Type *const first, const Type *const last) [inline], [static]`

Write a range between pointers to stream.

Writes each element of the range between the passed pointers to the stream and swap bytes if necessary.

Template Parameters

<i>Byteswap</i>	Swap endianness for writing if true.
<i>Type</i>	(Pointer type.)

Parameters

<i>in, out</i>	<i>stream</i>	Output stream.
<i>in</i>	<i>first</i>	Pointer to the beginning.
<i>in</i>	<i>last</i>	Pointer to the end.

Returns

True if the stream has no failure, false otherwise.

6.19.2.90 `template<bool Byteswap, typename Type , class > bool magrathea::DataHandler::rwrite (std::ostream & stream, const Type & first, const Type & last, typename std::iterator_traits< Type >::iterator_category * = nullptr) [inline], [static]`

Write a range between iterators to stream.

Writes each element of the range between the passed iterators to the stream and swap bytes if necessary.

Template Parameters

<i>Byteswap</i>	Swap endianness for writing if true.
<i>Type</i>	(Iterator type.)

Parameters

<i>in,out</i>	<i>stream</i>	Output stream.
<i>in</i>	<i>first</i>	Iterator to the beginning.
<i>in</i>	<i>last</i>	Iterator to the end.

Returns

True if the stream has no failure, false otherwise.

6.19.2.91 `template<bool Byteswap> char *& magrathea::DataHandler::rwrite (char *& buffer, const std::nullptr_t = nullptr, const std::nullptr_t = nullptr) [inline], [static]`

Write an empty range to buffer.

Function overload that does nothing.

Template Parameters

<i>Byteswap</i>	Swap endianness for writing if true.
-----------------	--------------------------------------

Parameters

<i>in,out</i>	<i>buffer</i>	Output buffer.
---------------	---------------	----------------

Returns

Pointer to the new position in the buffer.

6.19.2.92 `template<bool Byteswap, typename Type > char *& magrathea::DataHandler::rwrite (char *& buffer, const Type *const first, const Type *const last) [inline], [static]`

Write a range between pointers to buffer.

Writes each element of the range between the passed pointers to the buffer and swap bytes if necessary.

Template Parameters

<i>Byteswap</i>	Swap endianness for writing if true.
<i>Type</i>	(Pointer type.)

Parameters

<i>in,out</i>	<i>buffer</i>	Output buffer.
<i>in</i>	<i>first</i>	Pointer to the beginning.
<i>in</i>	<i>last</i>	Pointer to the end.

Returns

Pointer to the new position in the buffer.

6.19.2.93 `template<bool Byteswap, typename Type , class > char *& magrathea::DataHandler::rwrite (char *& buffer, const Type & first, const Type & last, typename std::iterator_traits< Type >::iterator_category * = nullptr) [inline], [static]`

Write a range between iterators to buffer.

Writes each element of the range between the passed iterators to the buffer and swap bytes if necessary.

Template Parameters

<i>Byteswap</i>	Swap endianness for writing if true.
<i>Type</i>	(Iterator type.)

Parameters

<i>in,out</i>	<i>buffer</i>	Output buffer.
<i>in</i>	<i>first</i>	Iterator to the beginning.
<i>in</i>	<i>last</i>	Iterator to the end.

Returns

Pointer to the new position in the buffer.

6.19.2.94 `bool magrathea::DataHandler::scan(std::istream & stream) [inline], [static]`

Scan nothing from stream.

Function overload that does nothing.

Parameters

<i>in,out</i>	<i>stream</i>	Input stream.
---------------	---------------	-------------------------------

Returns

True if the stream has no failure, false otherwise.

6.19.2.95 `bool magrathea::DataHandler::scan(std::istream & stream, std::tuple<> &) [inline], [static]`

Scan an empty tuple from stream.

Function overload that does nothing.

Parameters

<i>in,out</i>	<i>stream</i>	Input stream.
---------------	---------------	-------------------------------

Returns

True if the stream has no failure, false otherwise.

6.19.2.96 `template<typename Type > bool magrathea::DataHandler::scan(std::istream & stream, Type & variable) [inline], [static]`

Scan a single variable from stream.

Scans the next variable from the stream.

Template Parameters

<i>Type</i>	(Variable type.)
-------------	------------------

Parameters

<i>in,out</i>	<i>stream</i>	Input stream.
<i>out</i>	<i>variable</i>	Variable.

Returns

True if the stream has no failure, false otherwise.

6.19.2.97 template<typename Type , typename... Types> bool magrathea::DataHandler::scan (std::istream & *stream*, Type & *variable*, Types &... *variables*) [inline], [static]

Scan several variables from stream.

Scans the next variables from the stream.

Template Parameters

<i>Type</i>	(Variable type.)
<i>Types</i>	(Variadic types.)

Parameters

<i>in,out</i>	<i>stream</i>	Input stream.
<i>out</i>	<i>variable</i>	Variable.
<i>out</i>	<i>variables</i>	Variables.

Returns

True if the stream has no failure, false otherwise.

6.19.2.98 template<typename Type , std::size_t Size> bool magrathea::DataHandler::scan (std::istream & *stream*, std::array<Type, Size > & *container*) [inline], [static]

Scan an array from stream.

Scans each element of the passed array from the stream.

Template Parameters

<i>Type</i>	(Array type.)
<i>Size</i>	(Array size.)

Parameters

<i>in,out</i>	<i>stream</i>	Input stream.
<i>out</i>	<i>container</i>	Array.

Returns

True if the stream has no failure, false otherwise.

6.19.2.99 template<unsigned int Current, typename... Types> bool magrathea::DataHandler::scan (std::istream & *stream*, std::tuple<Types...> & *container*) [inline], [static]

Scan a tuple from stream.

Scans each element of the passed tuple from the stream.

Template Parameters

<i>Current</i>	(Current level of recursion.)
<i>Types</i>	(Variadic types.)

Parameters

<i>in, out</i>	<i>stream</i>	Input stream.
<i>out</i>	<i>container</i>	Tuple.

Returns

True if the stream has no failure, false otherwise.

6.19.2.100 template<typename... Types> constexpr unsigned int magrathea::DataHandler::size() [static]

Number of elements of several types.

Computes the number of elements of the passed types. Note that if the passed type is a templated type, like a tuple, the inner arguments are not extracted, and the result is simply the same as `sizeof...()`.

Returns

Number of elements.

6.19.2.101 constexpr unsigned int magrathea::DataHandler::size(const std::tuple<>&) [static]

Number of elements of an empty tuple.

Function overload that does nothing.

Returns

Number of elements.

6.19.2.102 template<typename Type > constexpr unsigned int magrathea::DataHandler::size(const Type & variable) [static]

Number of elements of a single variable.

Computes the number of elements of the passed variable.

Template Parameters

<i>Type</i>	(Variable type.)
-------------	------------------

Parameters

<i>in</i>	<i>variable</i>	Variable.
-----------	-----------------	-----------

Returns

Number of elements.

6.19.2.103 template<typename Type , typename... Types> constexpr unsigned int magrathea::DataHandler::size (const Type & *variable*, const Types &... *variables*) [static]

Number of elements of several variables.

Computes the number of elements of the passed variables.

Template Parameters

Type	(Variable type.)
Types	(Variadic types.)

Parameters

in	<i>variable</i>	Variable.
in	<i>variables</i>	Variables.

Returns

Number of elements.

6.19.2.104 template<typename Type , std::size_t Size> constexpr unsigned int magrathea::DataHandler::size (const std::array< Type, Size > & *container*) [static]

Number of elements of an array.

Computes the number of elements of the passed array.

Template Parameters

Type	(Array type.)
Size	(Array size.)

Parameters

in	<i>container</i>	Array.
----	------------------	--------

Returns

Number of elements.

6.19.2.105 template<unsigned int Current, typename... Types> constexpr unsigned int magrathea::DataHandler::size (const std::tuple< Types... > & *container*) [static]

Number of elements of a tuple.

Computes the number of elements of the passed tuple.

Template Parameters

Current	(Current level of recursion.)
Types	(Variadic types.)

Parameters

in	<i>container</i>	Tuple.
----	------------------	--------

Returns

Number of elements.

6.19.2.106 template<unsigned int Base, char Leading> std::string magrathea::DataHandler::stringify() [static]

Convert nothing to a string.

Function overload that does nothing.

Template Parameters

<i>Base</i>	Base representation for integers : 2, 8, 10 or 16.
<i>Leading</i>	Leading character for integers : 0, 9-13 or 32-126.

Returns

String conversion of arguments.

6.19.2.107 template<unsigned int Base, char Leading> std::string magrathea::DataHandler::stringify(const std::tuple<> &) [static]

Convert an empty tuple to a string.

Function overload that does nothing.

Template Parameters

<i>Base</i>	Base representation for integers : 2, 8, 10 or 16.
<i>Leading</i>	Leading character for integers : 0, 9-13 or 32-126.

Returns

String conversion of arguments.

6.19.2.108 template<unsigned int Base, char Leading, typename Type > std::string magrathea::DataHandler::stringify(const Type & variable) [static]

Convert a single variable to a string.

Converts the passed variable to a string. The base of conversion can be specified for integral numbers, and it is possible to fill-in the result with leading characters.

Template Parameters

<i>Base</i>	Base representation for integers : 2, 8, 10 or 16.
<i>Leading</i>	Leading character for integers : 0, 9-13 or 32-126.
<i>Type</i>	(Variable type.)

Parameters

<i>in</i>	<i>variable</i>	Variable.
-----------	-----------------	-----------

Returns

String conversion of arguments.

**6.19.2.109 template<unsigned int Base, char Leading, typename Type , typename... Types> std::string
magrathea::DataHandler::stringify (const Type & variable, const Types &... variables) [static]**

Convert several variables to a string.

Converts the passed variables to a string.

Template Parameters

<i>Base</i>	Base representation for integers : 2, 8, 10 or 16.
<i>Leading</i>	Leading character for integers : 0, 9-13 or 32-126.
<i>Type</i>	(Variable type.)
<i>Types</i>	(Variadic types.)

Parameters

<i>out</i>	<i>variable</i>	Variable.
<i>out</i>	<i>variables</i>	Variables.

Returns

String conversion of arguments.

**6.19.2.110 template<unsigned int Base, char Leading, typename Type , std::size_t Size> std::string
magrathea::DataHandler::stringify (const std::array<Type, Size > & container) [static]**

Convert an array to a string.

Converts each element of the passed array to a string.

Template Parameters

<i>Base</i>	Base representation for integers : 2, 8, 10 or 16.
<i>Leading</i>	Leading character for integers : 0, 9-13 or 32-126.
<i>Type</i>	(Array type.)
<i>Size</i>	(Array size.)

Parameters

<i>out</i>	<i>container</i>	Array.
------------	------------------	--------

Returns

String conversion of arguments.

6.19.2.111 template<unsigned int Base, char Leading, unsigned int Current, typename... Types> std::string magrathea::DataHandler::stringify (const std::tuple<Types...> & container) [static]

Convert a tuple to a string.

Converts each element of the passed tuple to a string.

Template Parameters

<i>Base</i>	Base representation for integers : 2, 8, 10 or 16.
<i>Leading</i>	Leading character for integers : 0, 9-13 or 32-126.
<i>Current</i>	(Current level of recursion.)
<i>Types</i>	(Variadic types.)

Parameters

<i>out</i>	<i>container</i>	Tuple.
------------	------------------	--------

Returns

String conversion of arguments.

6.19.2.112 template<typename Type > constexpr bool magrathea::DataHandler::tuple (const Type &) [static]

Is not a tuple.

Returns false as the passed argument is not a standard tuple.

Returns

Whether the argument is a standard tuple.

6.19.2.113 template<typename... Types> constexpr bool magrathea::DataHandler::tuple (const std::tuple<Types...> &) [static]

Is a tuple.

Returns true as the passed argument is a standard tuple.

Returns

Whether the argument is a standard tuple.

6.19.2.114 template<bool Byteswap> bool magrathea::DataHandler::write (std::ostream & stream) [inline], [static]

Write nothing to stream.

Function overload that does nothing.

Template Parameters

<i>Byteswap</i>	Swap endianness for writing if true.
-----------------	--------------------------------------

Parameters

<i>in,out</i>	<i>stream</i>	Output stream.
---------------	---------------	-----------------------

Returns

True if the stream has no failure, false otherwise.

6.19.2.115 `template<bool Byteswap> bool magrathea::DataHandler::write (std::ostream & stream, const std::tuple<> &) [inline], [static]`

Write an empty tuple to stream.

Function overload that does nothing.

Template Parameters

<i>Byteswap</i>	Swap endianness for writing if true.
-----------------	--------------------------------------

Parameters

<i>in,out</i>	<i>stream</i>	Output stream.
---------------	---------------	-----------------------

Returns

True if the stream has no failure, false otherwise.

6.19.2.116 `template<bool Byteswap, typename Type > bool magrathea::DataHandler::write (std::ostream & stream, const Type & variable) [inline], [static]`

Write a single variable to stream.

Writes the next variable to the stream and swap bytes if necessary.

Template Parameters

<i>Byteswap</i>	Swap endianness for writing if true.
<i>Type</i>	(Variable type.)

Parameters

<i>in,out</i>	<i>stream</i>	Output stream.
<i>in</i>	<i>variable</i>	Variable.

Returns

True if the stream has no failure, false otherwise.

6.19.2.117 `template<bool Byteswap, typename Type , typename... Types> bool magrathea::DataHandler::write (std::ostream & stream, const Type & variable, const Types &... variables) [inline], [static]`

Write several variables to stream.

Writes the next variables to the stream and swap bytes if necessary.

Template Parameters

<i>Byteswap</i>	Swap endianness for writing if true.
<i>Type</i>	(Variable type.)
<i>Types</i>	(Variadic types.)

Parameters

<i>in,out</i>	<i>stream</i>	Output stream.
<i>in</i>	<i>variable</i>	Variable.
<i>in</i>	<i>variables</i>	Variables.

Returns

True if the stream has no failure, false otherwise.

6.19.2.118 `template<bool Byteswap, typename Type , std::size_t Size> bool magrathea::DataHandler::write (std::ostream & stream, const std::array< Type, Size > & container) [inline], [static]`

Write an array to stream.

Writes each element of the passed array to the stream and swap bytes if necessary.

Template Parameters

<i>Byteswap</i>	Swap endianness for writing if true.
<i>Type</i>	(Array type.)
<i>Size</i>	(Array size.)

Parameters

<i>in,out</i>	<i>stream</i>	Output stream.
<i>in</i>	<i>container</i>	Array.

Returns

True if the stream has no failure, false otherwise.

6.19.2.119 `template<bool Byteswap, unsigned int Current, typename... Types> bool magrathea::DataHandler::write (std::ostream & stream, const std::tuple< Types... > & container) [inline], [static]`

Write a tuple to stream.

Writes each element of the passed tuple to the stream and swap bytes if necessary.

Template Parameters

<i>Byteswap</i>	Swap endianness for writing if true.
<i>Current</i>	(Current level of recursion.)
<i>Types</i>	(Variadic types.)

Parameters

<i>in, out</i>	<i>stream</i>	Output stream.
<i>in</i>	<i>container</i>	Tuple.

Returns

True if the stream has no failure, false otherwise.

6.19.2.120 template<bool Byteswap> char *& magrathea::DataHandler::write (char *& *buffer*) [inline], [static]

Write nothing to buffer.

Function overload that does nothing.

Template Parameters

<i>Byteswap</i>	Swap endianness for writing if true.
-----------------	--------------------------------------

Parameters

<i>in, out</i>	<i>buffer</i>	Output buffer.
----------------	---------------	-----------------------

Returns

Pointer to the new position in the buffer.

6.19.2.121 template<bool Byteswap> char *& magrathea::DataHandler::write (char *& *buffer*, const std::tuple<> &) [inline], [static]

Write an empty tuple to buffer.

Function overload that does nothing.

Template Parameters

<i>Byteswap</i>	Swap endianness for writing if true.
-----------------	--------------------------------------

Parameters

<i>in, out</i>	<i>buffer</i>	Output buffer.
----------------	---------------	-----------------------

Returns

Pointer to the new position in the buffer.

6.19.2.122 template<bool Byteswap, typename Type > char *& magrathea::DataHandler::write (char *& *buffer*, const Type & *variable*) [inline], [static]

Write a single variable to buffer.

Writes the next variable to the buffer and swap bytes if necessary.

Template Parameters

<i>Byteswap</i>	Swap endianness for writing if true.
<i>Type</i>	(Variable type.)

Parameters

<i>in,out</i>	<i>buffer</i>	Output buffer.
<i>in</i>	<i>variable</i>	Variable.

Returns

Pointer to the new position in the buffer.

6.19.2.123 `template<bool Byteswap, typename Type , typename... Types> char *& magrathea::DataHandler::write (char *& buffer, const Type & variable, const Types &... variables) [inline], [static]`

Write several variables to buffer.

Writes the next variables to the buffer and swap bytes if necessary.

Template Parameters

<i>Byteswap</i>	Swap endianness for writing if true.
<i>Type</i>	(Variable type.)
<i>Types</i>	(Variadic types.)

Parameters

<i>in,out</i>	<i>buffer</i>	Output buffer.
<i>in</i>	<i>variable</i>	Variable.
<i>in</i>	<i>variables</i>	Variables.

Returns

Pointer to the new position in the buffer.

6.19.2.124 `template<bool Byteswap, typename Type , std::size_t Size> char *& magrathea::DataHandler::write (char *& buffer, const std::array< Type, Size > & container) [inline], [static]`

Write an array to buffer.

Writes each element of the passed array to the buffer and swap bytes if necessary.

Template Parameters

<i>Byteswap</i>	Swap endianness for writing if true.
<i>Type</i>	(Array type.)
<i>Size</i>	(Array size.)

Parameters

<i>in,out</i>	<i>buffer</i>	Output buffer.
<i>in</i>	<i>container</i>	Array.

Returns

Pointer to the new position in the buffer.

6.19.2.125 template<bool Byteswap, unsigned int Current, typename... Types> char *& magrathea::DataHandler::write (char *& *buffer*, const std::tuple< Types... > & *container*) [inline], [static]

Write a tuple to buffer.

Writes each element of the passed tuple to the buffer and swap bytes if necessary.

Template Parameters

<i>Byteswap</i>	Swap endianness for writing if true.
<i>Current</i>	(Current level of recursion.)
<i>Types</i>	(Variadic types.)

Parameters

<i>in,out</i>	<i>buffer</i>	Output buffer.
<i>in</i>	<i>container</i>	Tuple.

Returns

Pointer to the new position in the buffer.

The documentation for this exception was generated from the following file:

- /data/home/mbreton/magrathea_pathfinder/src/magrathea/[datahandler.h](#)

6.20 magrathea::DataModel Exception Reference

Management of fundamental types representation.

```
#include <datamodel.h>
```

Public Member Functions

Lifecycle

- [DataModel](#) (const unsigned long long int source=0)
Explicit value constructor.
- [DataModel](#) (const bool big, const bool twos, const bool fieee754, const bool dieee754, const bool ldieee754, const unsigned int psize, const unsigned int bsize, const unsigned int csize, const unsigned int sisize, const unsigned int isize, const unsigned int lisize, const unsigned int llisize, const unsigned int fsize, const unsigned int dsize, const unsigned int ldsize)
Explicit detailed constructor.

Operators

- unsigned long long int [operator\(\)](#) ()
Access operator.
- bool [operator==](#) (const [DataModel](#) &rhs)
Equal to comparison.
- bool [operator!=](#) (const [DataModel](#) &rhs)
Not equal to comparison.

Assignment

- [DataModel](#) & [assign](#) (const [DataModel](#) &source)

- **Copy assignment.**
- **DataModel & assign** (const unsigned long long int source=0)
 - Value assignment.*
- **DataModel & assign** (const bool big, const bool twos, const bool fieee754, const bool dieee754, const bool ldieee754, const unsigned int psize, const unsigned int bsize, const unsigned int csize, const unsigned int sisize, const unsigned int isize, const unsigned int lisize, const unsigned int llisize, const unsigned int fsize, const unsigned int dsize, const unsigned int ldsiz)
 - Detailed assignment.*

Management

- unsigned int **size** () const
 - Get the size of the code.*
- const unsigned long long int & **data** () const
 - Access data.*
- **DataModel & clear** ()
 - Clear code.*
- **DataModel copy** () const
 - Copy.*
- template<typename Type = DataModel>
 Type cast () const
 - Cast.*
- bool **check** ()
 - Check if standard.*

Getters

- unsigned long long int **get** () const
 - Global getter.*
- bool **endianness** () const
 - Get endianness.*
- bool **complement** () const
 - Get complement.*
- template<typename Type , class = typename std::enable_if<(!std::is_pointer<Type>::value) && (std::is_floating_point<typename std::decay<Type>::type>::value)>::type>
 bool ieee754 () const
 - Get IEEE-754 compatibility.*
- template<typename Type , class = typename std::enable_if<(std::is_pointer<Type>::value) || (std::is_integral<typename std::decay<Type>::type>::value) || (std::is_floating_point<typename std::decay<Type>::type>::value)>::type>
 unsigned int **size** () const
 - Get type size.*

Setters

- **DataModel & set** (const bool big, const bool twos, const bool fieee754, const bool dieee754, const bool ldieee754, const unsigned int psize, const unsigned int bsize, const unsigned int csize, const unsigned int sisize, const unsigned int isize, const unsigned int lisize, const unsigned int llisize, const unsigned int fsize, const unsigned int dsize, const unsigned int ldsiz)
 - Global setter.*
- **DataModel & endianness** (const bool value)
 - Set endianness.*
- **DataModel & complement** (const bool value)
 - Set complement.*
- template<typename Type , class = typename std::enable_if<(!std::is_pointer<Type>::value) && (std::is_floating_point<typename std::decay<Type>::type>::value)>::type>
 DataModel & ieee754 (const bool value)
 - Set IEEE-754 compatibility.*
- template<typename Type , class = typename std::enable_if<(std::is_pointer<Type>::value) || (std::is_integral<typename std::decay<Type>::type>::value) || (std::is_floating_point<typename std::decay<Type>::type>::value)>::type>
 DataModel & size (const unsigned int value)
 - Set type size.*

Static Public Member Functions

Helpers

- template<typename Type , typename Decayed = typename std::decay<Type>::type, typename Signed = typename std::make_signed<typename std::conditional<(std::is_integral<Decayed>::value) && (!std::is_same<Decayed, bool>::value), Decayed, int>::type>::type, typename Fundamental = typename std::conditional<std::is_pointer<Type>::value, void*, typename std::conditional<(std::is_integral<Decayed>::value) && (!std::is_same<Decayed, bool>::value), typename std::conditional<std::is_same<Signed, signed char>::value, char, Signed>::type, Decayed>::type>::type, class = typename std::enable_if<(std::is_pointer<Fundamental>::value) || (std::is_integral<Fundamental>::value) || (std::is_floating_point<Fundamental>::value)>::type> static constexpr Fundamental fundamental ()
Get the associated fundamental type.
- template<typename Type , class = typename std::enable_if<(!std::is_pointer<Type>::value) && (std::is_floating_point<typename std::decay<Type>::type>::value)>::type> static constexpr bool control754 ()
Control IEEE-754 system compatibility.
- static constexpr bool control ()
Control if standard.

Predefined

- static const DataModel & system ()
System data model.

Test

- static int example ()
Example function.

Protected Attributes

Data members

- unsigned long long int _code
Internal encoding of data types.

Friends

Stream

- std::ostream & operator<< (std::ostream &lhs, const DataModel &rhs)
Output stream operator.

6.20.1 Detailed Description

Management of fundamental types representation.

Class to hold the data representation of fundamental types on a system. The information is encoded in an unsigned long long int in the following way where [BXbY-Z] means the information starts at the bit Y of the byte X and its size is Z bits :

- [B0b0-1] endianness
- [B0b1-1] complement
- [B1b0-1] float compatibility to the IEEE-754 binary32 representation
- [B1b1-1] double compatibility to the IEEE-754 binary64 representation

- [B1b2-1] long double compatibility to the IEEE-754 binary128 representation
- [B2b0-4] void* size
- [B2b4-4] bool size
- [B3b0-4] char size
- [B3b4-4] short int size
- [B4b0-4] int size
- [B4b4-4] long int size
- [B5b0-4] long long int size
- [B6b0-4] float size
- [B6b4-4] double size
- [B7b0-4] long double size

Specified sizes cannot be equal to zero and are defaulted to one. Furthermore, the IEEE-754 compatibility corresponds to correct byte size, IEC-559 compatibility, denormalization, correct radix and correct number of mantissa digits.

6.20.2 Constructor & Destructor Documentation

6.20.2.1 `magrathea::DataModel::DataModel(const unsigned long long int source = 0) [inline], [explicit]`

Explicit value constructor.

Explicitely constructs the data model from an unsigned long long int code.

Parameters

in	<code>source</code>	Code to be used for construction.
----	---------------------	-----------------------------------

6.20.2.2 `magrathea::DataModel::DataModel(const bool big, const bool twos, const bool fieee754, const bool dieee754, const bool ldieee754, const unsigned int psize, const unsigned int bsize, const unsigned int csize, const unsigned int sisize, const unsigned int isize, const unsigned int lisize, const unsigned int llisize, const unsigned int fsize, const unsigned int dsize, const unsigned int ldsiz) [inline], [explicit]`

Explicit detailed constructor.

Explicitely constructs the data model using all the needed values.

Parameters

in	<code>big</code>	False for little-endian, true for big-endian.
in	<code>twos</code>	True for two's complement, false otherwise.
in	<code>fieee754</code>	IEEE-754 binary32 compatibility of float.
in	<code>dieee754</code>	IEEE-754 binary64 compatibility of double.
in	<code>ldieee754</code>	IEEE-754 binary128 compatibility of long double.
in	<code>psize</code>	Byte size of pointers.
in	<code>bsize</code>	Byte size of bool.
in	<code>csize</code>	Byte size of char.
in	<code>sisize</code>	Byte size of short int.
in	<code>isize</code>	Byte size of int.
in	<code>lisize</code>	Byte size of long int.
in	<code>llisize</code>	Byte size of long long int.

in	<i>fsize</i>	Byte size of float.
in	<i>dsize</i>	Byte size of double.
in	<i>lsize</i>	Byte size of long double.

6.20.3 Member Function Documentation

6.20.3.1 DataModel & magrathea::DataModel::assign (const DataModel & source) [inline]

Copy assignment.

Assign the code from another data model.

Parameters

in	<i>source</i>	Source of the copy.
----	---------------	---------------------

Returns

Self reference.

6.20.3.2 DataModel & magrathea::DataModel::assign (const unsigned long long int source = 0) [inline]

Value assignment.

Assigns a code to the data model.

Parameters

in	<i>source</i>	Source of the copy.
----	---------------	---------------------

Returns

Self reference.

6.20.3.3 DataModel & magrathea::DataModel::assign (const bool *big*, const bool *twos*, const bool *fieee754*, const bool *dieee754*, const bool *ldieee754*, const unsigned int *psize*, const unsigned int *bsize*, const unsigned int *csize*, const unsigned int *ssize*, const unsigned int *isize*, const unsigned int *lsize*, const unsigned int *llsize*, const unsigned int *fsize*, const unsigned int *dsize*, const unsigned int *lsize*) [inline]

Detailed assignment.

Assigns the contents of the data model using all the needed values.

Parameters

in	<i>big</i>	False for little-endian, true for big-endian.
in	<i>twos</i>	True for two's complement, false otherwise.
in	<i>fieee754</i>	IEEE-754 binary32 compatibility of float.
in	<i>dieee754</i>	IEEE-754 binary64 compatibility of double.
in	<i>ldieee754</i>	IEEE-754 binary128 compatibility of long double.
in	<i>psize</i>	Byte size of pointers.
in	<i>bsize</i>	Byte size of bool.
in	<i>csize</i>	Byte size of char.
in	<i>ssize</i>	Byte size of short int.
in	<i>isize</i>	Byte size of int.

in	<i>lsize</i>	Byte size of long int.
in	<i>llsize</i>	Byte size of long long int.
in	<i>fsize</i>	Byte size of float.
in	<i>dsize</i>	Byte size of double.
in	<i>ldsize</i>	Byte size of long double.

Returns

Self reference.

6.20.3.4 template<typename Type > Type magrathea::DataModel::cast() const [inline]

Cast.

Returns a copy of the data model casted to the provided type.

Template Parameters

Type	Data type.
------	------------

Returns

Casted copy.

6.20.3.5 bool magrathea::DataModel::check() [inline]

Check if standard.

Checks whether the data model is a standard one. It means that it has all the following properties :

- it uses two's complement
- float and double are IEEE-754 compliant
- pointer size is 4 or 8
- bool size is 1
- char size is 1
- short int size is 2
- int size is 4
- long int size is 4 or 8
- long long int size is 8
- float size is 4
- double size is 8
- long double size is 8, 10, 12 or 16

Returns

True if the data model is standard, false if not.

6.20.3.6 DataModel & magrathea::DataModel::clear () [inline]

Clear code.

Clears the whole contents and sets the flags to zero, and the sizes to one.

Returns

Self reference.

6.20.3.7 bool magrathea::DataModel::complement () const [inline]

Get complement.

Returns whether the system use two's complement encoding or not according to the data model.

Returns

True for two's complement, false otherwise.

6.20.3.8 DataModel & magrathea::DataModel::complement (const bool value) [inline]

Set complement.

Sets the complement of the data model.

Parameters

in	value	True for two's complement, false otherwise.
----	-------	---

Returns

Self reference.

6.20.3.9 constexpr bool magrathea::DataModel::control () [static]

Control if standard.

Controls whether the system data model is a standard one. It means that it has all the following properties :

- it uses two's complement
- float and double are IEEE-754 compliant
- long double is IEC-559 compliant, can be denormalized and has a binary radix
- pointer size is 4 or 8
- bool size is 1
- char size is 1
- short int size is 2
- int size is 4
- long int size is 4 or 8
- long long int size is 8

- float size is 4
- double size is 8
- long double size is 8, 10, 12 or 16

Returns

True if the data model is standard, false if not.

6.20.3.10 template<typename Type , class > constexpr bool magrathea::DataModel::control754() [static]

Control IEEE-754 system compatibility.

Control whether the floating-point type is compatible with the IEEE-754 standard on the current architecture : correct byte size, IEC-559 compatibility, denormalization, correct radix and correct number of mantissa digits.

Template Parameters

Type	Floating-point type.
------	----------------------

Returns

True if the type is compliant to IEEE-754, false otherwise.

6.20.3.11 DataModel magrathea::DataModel::copy() const [inline]

Copy.

Returns a copy of the data model.

Returns

Copy.

6.20.3.12 const unsigned long long int & magrathea::DataModel::data() const [inline]

Access data.

Returns a constant reference to the internal underlying data which is the data model code.

Returns

Immutable reference to the code.

6.20.3.13 bool magrathea::DataModel::endianness() const [inline]

Get endianness.

Returns the endianness from data model.

Returns

False for little-endian, true for big-endian.

6.20.3.14 DataModel & magrathea::DataModel::endianness (const bool value) [inline]

Set endianness.

Sets the endianness of the data model.

Parameters

in	value	False for little-endian, true for big-endian.
----	-------	---

Returns

Self reference.

6.20.3.15 int magrathea::DataModel::example () [static]

Example function.

Tests and demonstrates the use of [DataModel](#).

Returns

0 if no error.

6.20.3.16 template<typename Type , typename Decayed , typename Signed , typename Fundamental , class > constexpr Fundamental magrathea::DataModel::fundamental () [static]

Get the associated fundamental type.

Converts the type passed as first template argument to an associated fundamental type :

- `void*` if the type is a pointer
- `bool` if the type is a cv-qualified boolean
- `char` if the type is a cv-qualified plain, signed or unsigned character
- `signed decayed type` if the type is an integral type
- `decayed type` if the type is a floating-point type

Template Parameters

Type	Type to convert.
Decayed	(Decayed version of the type.)
Signed	(Signed version of the type.)
Fundamental	(Fundamental version of the type.)

Returns

Default value of the fundamental type.

6.20.3.17 unsigned long long int magrathea::DataModel::get () const [inline]

Global getter.

Returns a copy of the underlying complete code of the data model.

Returns

Copy of the code.

6.20.3.18 template<typename Type , class > bool magrathea::DataModel::ieee754() const [inline]

Get IEEE-754 compatibility.

Returns whether the specified floating-point type is compliant to the IEEE-754 standard according to the data model.

Template Parameters

Type	Floating-point type.
------	----------------------

Returns

True if compliant, false otherwise.

6.20.3.19 template<typename Type , class > DataModel & magrathea::DataModel::ieee754(const bool value) [inline]

Set IEEE-754 compatibility.

Sets whether the specified floating-point type is compliant to the IEEE-754 standard.

Template Parameters

Type	Floating-point type.
------	----------------------

Parameters

in	value	True if compliant, false otherwise.
----	-------	-------------------------------------

Returns

Self reference.

6.20.3.20 bool magrathea::DataModel::operator!= (const DataModel & rhs) [inline]

Not equal to comparison.

Compares two data models for difference.

Returns

True if the two data models are not equal, false otherwise.

6.20.3.21 unsigned long long int magrathea::DataModel::operator() () [inline]

Access operator.

Returns a copy of the underlying complete code of the data model.

Returns

Copy of the code.

6.20.3.22 bool magrathea::DataModel::operator== (const DataModel & rhs) [inline]

Equal to comparison.

Compares two data models for equality.

Returns

True if the two data models are equal, false otherwise.

6.20.3.23 DataModel & magrathea::DataModel::set (const bool big, const bool twos, const bool fieee754, const bool dieee754, const bool ldieee754, const unsigned int psize, const unsigned int bsize, const unsigned int csize, const unsigned int sisize, const unsigned int isize, const unsigned int lisize, const unsigned int llisize, const unsigned int fsize, const unsigned int dsize, const unsigned int ldsiz) [inline]

Global setter.

Sets the content using all the needed values.

Parameters

in	<i>big</i>	False for little-endian, true for big-endian.
in	<i>twos</i>	True for two's complement, false otherwise.
in	<i>fieee754</i>	IEEE-754 binary32 compatibility of float.
in	<i>dieee754</i>	IEEE-754 binary64 compatibility of double.
in	<i>ldieee754</i>	IEEE-754 binary128 compatibility of long double.
in	<i>psize</i>	Byte size of pointers.
in	<i>bsize</i>	Byte size of bool.
in	<i>csize</i>	Byte size of char.
in	<i>sisize</i>	Byte size of short int.
in	<i>isize</i>	Byte size of int.
in	<i>lisize</i>	Byte size of long int.
in	<i>llisize</i>	Byte size of long long int.
in	<i>fsize</i>	Byte size of float.
in	<i>dsize</i>	Byte size of double.
in	<i>ldsiz</i>	Byte size of long double.

Returns

Self reference.

6.20.3.24 unsigned int magrathea::DataModel::size () const [inline]

Get the size of the code.

Get type size.

Returns the result of the `sizeof` operator on the underlying code.

Returns

Size of the code in bytes.

Returns the size of the provided fundamental type according to the data model.

Template Parameters

Type	Pointer, integral or floating-point type.
------	---

Returns

Type size in bytes.

6.20.3.25 template<typename Type , class = typename std::enable_if<(std::is_pointer<Type>::value) || (std::is_integral<typename std::decay<Type>::type>::value) || (std::is_floating_point<typename std::decay<Type>::type>::value)>::type> unsigned int magrathea::DataModel::size () const [inline]

6.20.3.26 template<typename Type , class > DataModel & magrathea::DataModel::size (const unsigned int value) [inline]

Set type size.

Sets the size of the provided fundamental type of the data model.

Template Parameters

Type	Pointer, integral or floating-point type.
------	---

Parameters

in	value	Type size in bytes.
----	-------	---------------------

Returns

Self reference.

6.20.3.27 const DataModel & magrathea::DataModel::system () [inline], [static]

System data model.

Returns an immutable reference to singleton representing the data model of the current architecture.

Returns

Immutable reference to system data model singleton.

6.20.4 Friends And Related Function Documentation

6.20.4.1 std::ostream& operator<< (std::ostream & lhs, const DataModel & rhs) [friend]

[Output](#) stream operator.

Prints out the data model.

Parameters

in,out	lhs	Left-hand side stream.
in	rhs	Right-hand side data model.

Returns

[Output](#) stream.

6.20.5 Member Data Documentation

6.20.5.1 unsigned long long int magrathea::DataModel::_code [protected]

Internal encoding of data types.

The documentation for this exception was generated from the following file:

• /data/home/mbreton/mgrathea_pathfinder/src/mgrathea/[datamodel.h](#)

6.21 mgrathea::DataSize Exception Reference

[Wrapper](#) of binary data size and manager of unit conversion.

```
#include <datasize.h>
```

Public Member Functions

Lifecycle

- `constexpr DataSize ()`
Implicit constexpr empty constructor.
- `constexpr DataSize (const long long int amount)`
Explicit constexpr value constructor.

Operators

- `template<typename Type >`
`DataSize & operator= (const Type &rhs)`
Conversion assignment operator.
- `DataSize & operator= (const DataSize &rhs)`
Copy assignment operator.
- `unsigned long long int operator() ()`
Conversion operator.

Management

- `constexpr const long long int & data () const`
Access data.
- `constexpr DataSize copy () const`
Copy.
- `template<typename Type = DataSize>`
`constexpr Type cast () const`
Cast.

Getters

- `bool constexpr valid () const`
Get the validity of the data size.
- `bool constexpr empty () const`
Get whether the size is not null.
- `template<typename Type = unsigned long long int>`
`constexpr Type bytes () const`

- template<typename Type = unsigned long long int>
constexpr Type kibibytes () const
Get data size in kibibytes.
- template<typename Type = unsigned long long int>
constexpr Type mebibytes () const
Get data size in mebibytes.
- template<typename Type = unsigned long long int>
constexpr Type gibibytes () const
Get data size in gibibytes.
- template<typename Type = unsigned long long int>
constexpr Type tebibytes () const
Get data size in tebibytes.
- template<typename Type = unsigned long long int>
constexpr Type pebibytes () const
Get data size in pebibytes.
- template<typename Type = unsigned long long int>
constexpr Type exbibytes () const
Get data size in exbibytes.
- template<typename Type = unsigned long long int>
constexpr Type kilobytes () const
Get data size in kilobytes.
- template<typename Type = unsigned long long int>
constexpr Type megabytes () const
Get data size in megabytes.
- template<typename Type = unsigned long long int>
constexpr Type gigabytes () const
Get data size in gigabytes.
- template<typename Type = unsigned long long int>
constexpr Type terabytes () const
Get data size in terabytes.
- template<typename Type = unsigned long long int>
constexpr Type petabytes () const
Get data size in petabytes.
- template<typename Type = unsigned long long int>
constexpr Type exabytes () const
Get data size in exabytes.
- template<typename Type = unsigned long long int>
constexpr Type bits () const
Get data size in bits.

Setters

- **DataSize & valid (const bool ok)**
Set valid status.
- **DataSize & empty (const bool ok)**
Set empty status.
- template<typename Type >
DataSize & bytes (const Type &amount)
Set data size in bytes.
- template<typename Type >
DataSize & kibibytes (const Type &amount)
Set data size in kibibytes.
- template<typename Type >
DataSize & mebibytes (const Type &amount)
Set data size in mebibytes.
- template<typename Type >
DataSize & gibibytes (const Type &amount)
Set data size in gibibytes.

- template<typename Type >
DataSize & tebibytes (const Type &amount)
Set data size in tebibytes.
- template<typename Type >
DataSize & pebibytes (const Type &amount)
Set data size in pebibytes.
- template<typename Type >
DataSize & exbibytes (const Type &amount)
Set data size in exbibytes.
- template<typename Type >
DataSize & kilobytes (const Type &amount)
Set data size in kilobytes.
- template<typename Type >
DataSize & megabytes (const Type &amount)
Set data size in megabytes.
- template<typename Type >
DataSize & gigabytes (const Type &amount)
Set data size in gigabytes.
- template<typename Type >
DataSize & terabytes (const Type &amount)
Set data size in terabytes.
- template<typename Type >
DataSize & petabytes (const Type &amount)
Set data size in petabytes.
- template<typename Type >
DataSize & exabytes (const Type &amount)
Set data size in exabytes.
- template<typename Type >
DataSize & bits (const Type &amount)
Set data size in bits.

Static Public Member Functions

Predefined

- template<typename Type = unsigned long long int>
static constexpr DataSize byte (const Type &amount=Type(1))
Predefined data size in bytes.
- template<typename Type = unsigned long long int>
static constexpr DataSize kibi (const Type &amount=Type(1))
Predefined data size in kibibytes.
- template<typename Type = unsigned long long int>
static constexpr DataSize mebi (const Type &amount=Type(1))
Predefined data size in mebibytes.
- template<typename Type = unsigned long long int>
static constexpr DataSize gibi (const Type &amount=Type(1))
Predefined data size in gibibytes.
- template<typename Type = unsigned long long int>
static constexpr DataSize tebi (const Type &amount=Type(1))
Predefined data size in tebibytes.
- template<typename Type = unsigned long long int>
static constexpr DataSize pebi (const Type &amount=Type(1))
Predefined data size in pebibytes.
- template<typename Type = unsigned long long int>
static constexpr DataSize exbi (const Type &amount=Type(1))
Predefined data size in exbibytes.
- template<typename Type = unsigned long long int>
static constexpr DataSize kilo (const Type &amount=Type(1))

- *Predefined data size in kilobytes.*
- template<typename Type = unsigned long long int>
static constexpr **DataSize mega** (const Type &amount=Type(1))
Predefined data size in megabytes.
- template<typename Type = unsigned long long int>
static constexpr **DataSize giga** (const Type &amount=Type(1))
Predefined data size in gigabytes.
- template<typename Type = unsigned long long int>
static constexpr **DataSize tera** (const Type &amount=Type(1))
Predefined data size in terabytes.
- template<typename Type = unsigned long long int>
static constexpr **DataSize peta** (const Type &amount=Type(1))
Predefined data size in petabytes.
- template<typename Type = unsigned long long int>
static constexpr **DataSize exa** (const Type &amount=Type(1))
Predefined data size in exabytes.
- template<typename Type = unsigned long long int>
static constexpr **DataSize bit** (const Type &amount=Type(8))
Predefined data size in bits.

Test

- static int **example** ()
Example function.

Protected Attributes

Data members

- long long int **_size**
Data size in bytes.

Friends

Stream

- std::ostream & **operator<<** (std::ostream &lhs, const **DataSize** &rhs)
Output stream operator.

6.21.1 Detailed Description

[Wrapper](#) of binary data size and manager of unit conversion.

Class to hold a byte count to represent the size of a file or a chunk of memory. The byte count is stored internally into a long long integer whose negative values are associated with a void data size like a non-existing file.

6.21.2 Constructor & Destructor Documentation

6.21.2.1 constexpr magrathea::DataSize::DataSize ()

Implicit constexpr empty constructor.

Implicitly constructs the data size and set it to an invalid size.

6.21.2.2 `constexpr magrathea::DataSize::DataSize(const long long int amount) [explicit]`

Explicit `constexpr` value constructor.

Explicitely constructs the data size from a long long integer.

Parameters

<code>in</code>	<code>amount</code>	Value to be used for construction.
-----------------	---------------------	------------------------------------

6.21.3 Member Function Documentation

6.21.3.1 `template<typename Type> constexpr DataSize magrathea::DataSize::bit(const Type & amount = Type(8)) [static]`

Predefined data size in bits.

Constructs and returns a data size based on an amount of bits. If the number of provided bits is not divisible by 8, an extra byte is added.

Template Parameters

<code>Type</code>	(Data type.)
-------------------	--------------

Parameters

<code>in</code>	<code>amount</code>	Amount of bits.
-----------------	---------------------	-----------------

Returns

Copy of the data size.

6.21.3.2 `template<typename Type> constexpr Type magrathea::DataSize::bits() const`

Get data size in bits.

Returns the data size in bits in the provided type.

Template Parameters

<code>Type</code>	Data type.
-------------------	------------

Returns

The amount of bits.

6.21.3.3 `template<typename Type> DataSize & magrathea::DataSize::bits(const Type & amount) [inline]`

Set data size in bits.

Sets the current data size in bits. If the number of provided bits is not divisible by 8, an extra byte is added.

Template Parameters

<code>Type</code>	(Data type.)
-------------------	--------------

Parameters

<code>in</code>	<code>amount</code>	Amount of bits.
-----------------	---------------------	-----------------

Returns

Self reference.

6.21.3.4 template<typename Type > constexpr DataSize magrathea::DataSize::byte(const Type & *amount* = Type(1)) [static]

Predefined data size in bytes.

Constructs and returns a data size based on an amount of bytes.

Template Parameters

<code>Type</code>	(Data type.)
-------------------	--------------

Parameters

<code>in</code>	<code>amount</code>	Amount of bytes.
-----------------	---------------------	------------------

Returns

Copy of the data size.

6.21.3.5 template<typename Type > constexpr Type magrathea::DataSize::bytes() const

Get data size in bytes.

Returns the data size in bytes in the provided type.

Template Parameters

<code>Type</code>	Data type.
-------------------	------------

Returns

The amount of bytes.

6.21.3.6 template<typename Type > DataSize & magrathea::DataSize::bytes(const Type & *amount*) [inline]

Set data size in bytes.

Sets the current data size in bytes.

Template Parameters

<code>Type</code>	(Data type.)
-------------------	--------------

Parameters

<code>in</code>	<code>amount</code>	Amount of bytes.
-----------------	---------------------	------------------

Returns

Self reference.

6.21.3.7 template<typename Type > constexpr Type magrathea::DataSize::cast () const**Cast.**

Returns a copy of the data size casted to the provided type. If the size is undefined and the type has a signaling NaN, then this signaling NaN is returned.

Template Parameters

Type	Data type.
------	------------

Returns

Copy.

6.21.3.8 constexpr DataSize magrathea::DataSize::copy () const**Copy.**

Returns a copy of the data size.

Returns

Copy.

6.21.3.9 constexpr const long long int & magrathea::DataSize::data () const**Access data.**

Returns a constant reference to the internal underlying data which is the size in bytes as a long long integer, negative if undefined.

Returns

Const reference to the data.

6.21.3.10 constexpr bool magrathea::DataSize::empty () const**Get whether the size is not null.**

Returns true if the size is equal to zero or if data is not valid.

Returns

Whether the size is not strictly greater than zero.

6.21.3.11 DataSize & magrathea::DataSize::empty (const bool ok) [inline]**Set empty status.**

Sets the size to zero if the argument is true, change it to the maximum between the current size and one byte if false.

Parameters

in	<i>ok</i>	Status.
----	-----------	---------

Returns

Self reference.

6.21.3.12 template<typename Type > constexpr DataSize magrathea::DataSize::exa (const Type & *amount* = Type(1)) [static]

Predefined data size in exabytes.

Constructs and returns a data size based on an amount of exabytes.

Template Parameters

Type	(Data type.)
------	--------------

Parameters

in	<i>amount</i>	Amount of exabytes.
----	---------------	---------------------

Returns

Copy of the data size.

6.21.3.13 template<typename Type > constexpr Type magrathea::DataSize::exabytes () const

Get data size in exabytes.

Returns the data size in exabytes in the provided type.

Template Parameters

Type	Data type.
------	------------

Returns

The amount of exabytes.

6.21.3.14 template<typename Type > DataSize & magrathea::DataSize::exabytes (const Type & *amount*) [inline]

Set data size in exabytes.

Sets the current data size in exabytes.

Template Parameters

Type	(Data type.)
------	--------------

Parameters

in	<i>amount</i>	Amount of exabytes.
----	---------------	---------------------

Returns

Self reference.

6.21.3.15 int magrathea::DataSize::example() [static]

Example function.

Tests and demonstrates the use of [DataSize](#).

Returns

0 if no error.

6.21.3.16 template<typename Type> constexpr DataSize magrathea::DataSize::exbi(const Type & amount = Type(1)) [static]

Predefined data size in exbibytes.

Constructs and returns a data size based on an amount of exbibytes.

Template Parameters

Type	(Data type.)
------	--------------

Parameters

in	amount	Amount of exbibytes.
----	--------	----------------------

Returns

Copy of the data size.

6.21.3.17 template<typename Type> constexpr Type magrathea::DataSize::exbibytes() const

Get data size in exbibytes.

Returns the data size in exbibytes in the provided type.

Template Parameters

Type	Data type.
------	------------

Returns

The amount of exbibytes.

6.21.3.18 template<typename Type> DataSize & magrathea::DataSize::exbibytes(const Type & amount) [inline]

Set data size in exbibytes.

Sets the current data size in exbibytes.

Template Parameters

Type	(Data type.)
------	--------------

Parameters

in	amount	Amount of exbibytes.
----	--------	----------------------

Returns

Self reference.

6.21.3.19 template<typename Type > constexpr DataSize magrathea::DataSize::gibi (const Type & amount = Type(1)) [static]

Predefined data size in gibibytes.

Constructs and returns a data size based on an amount of gibibytes.

Template Parameters

Type	(Data type.)
------	--------------

Parameters

in	amount	Amount of gibibytes.
----	--------	----------------------

Returns

Copy of the data size.

6.21.3.20 template<typename Type > constexpr Type magrathea::DataSize::gibibytes () const

Get data size in gibibytes.

Returns the data size in gibibytes in the provided type.

Template Parameters

Type	Data type.
------	------------

Returns

The amount of gibibytes.

6.21.3.21 template<typename Type > DataSize & magrathea::DataSize::gibibytes (const Type & amount) [inline]

Set data size in gibibytes.

Sets the current data size in gibibytes.

Template Parameters

Type	(Data type.)
------	--------------

Parameters

in	amount	Amount of gibibytes.
----	--------	----------------------

Returns

Self reference.

6.21.3.22 `template<typename Type> constexpr DataSize magrathea::DataSize::giga (const Type & amount = Type(1)) [static]`

Predefined data size in gigabytes.

Constructs and returns a data size based on an amount of gigabytes.

Template Parameters

Type	(Data type.)
------	--------------

Parameters

in	amount	Amount of gigabytes.
----	--------	----------------------

Returns

Copy of the data size.

6.21.3.23 `template<typename Type> constexpr Type magrathea::DataSize::gigabytes () const`

Get data size in gigabytes.

Returns the data size in gigabytes in the provided type.

Template Parameters

Type	Data type.
------	------------

Returns

The amount of gigabytes.

6.21.3.24 `template<typename Type> DataSize & magrathea::DataSize::gigabytes (const Type & amount) [inline]`

Set data size in gigabytes.

Sets the current data size in gigabytes.

Template Parameters

Type	(Data type.)
------	--------------

Parameters

in	amount	Amount of gigabytes.
----	--------	----------------------

Returns

Self reference.

6.21.3.25 `template<typename Type > constexpr DataSize magrathea::DataSize::kibi (const Type & amount = Type(1)) [static]`

Predefined data size in kibibytes.

Constructs and returns a data size based on an amount of kibibytes.

Template Parameters

<code>Type</code>	(Data type.)
-------------------	--------------

Parameters

<code>in</code>	<code>amount</code>	Amount of kibibytes.
-----------------	---------------------	----------------------

Returns

Copy of the data size.

6.21.3.26 `template<typename Type > constexpr Type magrathea::DataSize::kibibytes () const`

Get data size in kibibytes.

Returns the data size in kibibytes in the provided type.

Template Parameters

<code>Type</code>	Data type.
-------------------	------------

Returns

The amount of kibibytes.

6.21.3.27 `template<typename Type > DataSize & magrathea::DataSize::kibibytes (const Type & amount) [inline]`

Set data size in kibibytes.

Sets the current data size in kibibytes.

Template Parameters

<code>Type</code>	(Data type.)
-------------------	--------------

Parameters

<code>in</code>	<code>amount</code>	Amount of kibibytes.
-----------------	---------------------	----------------------

Returns

Self reference.

6.21.3.28 `template<typename Type > constexpr DataSize magrathea::DataSize::kilo (const Type & amount = Type(1)) [static]`

Predefined data size in kilobytes.

Constructs and returns a data size based on an amount of kilobytes.

Template Parameters

Type	(Data type.)
------	--------------

Parameters

in	amount	Amount of kilobytes.
----	--------	----------------------

Returns

Copy of the data size.

6.21.3.29 template<typename Type > constexpr Type magrathea::DataSize::kilobytes () const

Get data size in kilobytes.

Returns the data size in kilobytes in the provided type.

Template Parameters

Type	Data type.
------	------------

Returns

The amount of kilobytes.

6.21.3.30 template<typename Type > DataSize & magrathea::DataSize::kilobytes (const Type & amount) [inline]

Set data size in kilobytes.

Sets the current data size in kilobytes.

Template Parameters

Type	(Data type.)
------	--------------

Parameters

in	amount	Amount of kilobytes.
----	--------	----------------------

Returns

Self reference.

6.21.3.31 template<typename Type > constexpr DataSize magrathea::DataSize::mebi (const Type & amount = Type(1)) [static]

Predefined data size in mebibytes.

Constructs and returns a data size based on an amount of mebibytes.

Template Parameters

Type	(Data type.)
------	--------------

Parameters

in	<i>amount</i>	Amount of mebibytes.
----	---------------	----------------------

Returns

Copy of the data size.

6.21.3.32 template<typename Type > constexpr Type magrathea::DataSize::mebibytes() const

Get data size in mebibytes.

Returns the data size in mebibytes in the provided type.

Template Parameters

<i>Type</i>	Data type.
-------------	------------

Returns

The amount of mebibytes.

6.21.3.33 template<typename Type > DataSize & magrathea::DataSize::mebibytes(const Type & amount) [inline]

Set data size in mebibytes.

Sets the current data size in mebibytes.

Template Parameters

<i>Type</i>	(Data type.)
-------------	--------------

Parameters

in	<i>amount</i>	Amount of mebibytes.
----	---------------	----------------------

Returns

Self reference.

6.21.3.34 template<typename Type > constexpr DataSize magrathea::DataSize::mega(const Type & amount = Type(1)) [static]

Predefined data size in megabytes.

Constructs and returns a data size based on an amount of megabytes.

Template Parameters

<i>Type</i>	(Data type.)
-------------	--------------

Parameters

in	<i>amount</i>	Amount of megabytes.
----	---------------	----------------------

Returns

Copy of the data size.

6.21.3.35 template<typename Type > constexpr Type magrathea::DataSize::megabytes() const

Get data size in megabytes.

Returns the data size in megabytes in the provided type.

Template Parameters

Type	Data type.
------	------------

Returns

The amount of megabytes.

6.21.3.36 template<typename Type > DataSize & magrathea::DataSize::megabytes(const Type & amount) [inline]

Set data size in megabytes.

Sets the current data size in megabytes.

Template Parameters

Type	(Data type.)
------	--------------

Parameters

in	amount	Amount of megabytes.
----	--------	----------------------

Returns

Self reference.

6.21.3.37 unsigned long long int magrathea::DataSize::operator()() [inline]

Conversion operator.

Returns the number of bytes or throw an error if the size is not defined.

Returns

The size in bytes.

Exceptions

std::underflow_error	Undefined size.
----------------------	-----------------

6.21.3.38 template<typename Type > DataSize & magrathea::DataSize::operator=(const Type & rhs) [inline]

Conversion assignment operator.

Convert the provided amount of bytes to a data size.

Template Parameters

Type	(Data type.)
------	--------------

Parameters

in	rhs	Right-hand side.
----	-----	------------------

Returns

Self reference.

6.21.3.39 DataSize & magrathea::DataSize::operator= (const DataSize & rhs) [inline]

Copy assignment operator.

Copies the contents of another data size.

Parameters

in	rhs	Right-hand side.
----	-----	------------------

Returns

Self reference.

6.21.3.40 template<typename Type > constexpr DataSize magrathea::DataSize::pebi (const Type & amount = Type(1)) [static]

Predefined data size in pebibytes.

Constructs and returns a data size based on an amount of pebibytes.

Template Parameters

Type	(Data type.)
------	--------------

Parameters

in	amount	Amount of pebibytes.
----	--------	----------------------

Returns

Copy of the data size.

6.21.3.41 template<typename Type > constexpr Type magrathea::DataSize::pebibytes () const

Get data size in pebibytes.

Returns the data size in pebibytes in the provided type.

Template Parameters

Type	Data type.
------	------------

Returns

The amount of pebibytes.

6.21.3.42 template<typename Type > DataSize & magrathea::DataSize::pebibytes (const Type & *amount*) [inline]

Set data size in pebibytes.

Sets the current data size in pebibytes.

Template Parameters

Type	(Data type.)
------	--------------

Parameters

in	amount	Amount of pebibytes.
----	--------	----------------------

Returns

Self reference.

6.21.3.43 template<typename Type > constexpr DataSize magrathea::DataSize::peta (const Type & *amount* = Type(1)) [static]

Predefined data size in petabytes.

Constructs and returns a data size based on an amount of petabytes.

Template Parameters

Type	(Data type.)
------	--------------

Parameters

in	amount	Amount of petabytes.
----	--------	----------------------

Returns

Copy of the data size.

6.21.3.44 template<typename Type > constexpr Type magrathea::DataSize::petabytes () const

Get data size in petabytes.

Returns the data size in petabytes in the provided type.

Template Parameters

Type	Data type.
------	------------

Returns

The amount of petabytes.

6.21.3.45 template<typename Type > DataSize & magrathea::DataSize::petabytes (const Type & *amount*) [inline]

Set data size in petabytes.

Sets the current data size in petabytes.

Template Parameters

Type	(Data type.)
------	--------------

Parameters

in	amount	Amount of petabytes.
----	--------	----------------------

Returns

Self reference.

6.21.3.46 template<typename Type > constexpr DataSize magrathea::DataSize::tebi (const Type & *amount* = Type(1)) [static]

Predefined data size in tebibytes.

Constructs and returns a data size based on an amount of tebibytes.

Template Parameters

Type	(Data type.)
------	--------------

Parameters

in	amount	Amount of tebibytes.
----	--------	----------------------

Returns

Copy of the data size.

6.21.3.47 template<typename Type > constexpr Type magrathea::DataSize::tebibytes () const

Get data size in tebibytes.

Returns the data size in tebibytes in the provided type.

Template Parameters

Type	Data type.
------	------------

Returns

The amount of tebibytes.

6.21.3.48 template<typename Type > DataSize & magrathea::DataSize::tebibytes (const Type & *amount*) [inline]

Set data size in tebibytes.

Sets the current data size in tebibytes.

Template Parameters

Type	(Data type.)
------	--------------

Parameters

in	amount	Amount of tebibytes.
----	--------	----------------------

Returns

Self reference.

6.21.3.49 template<typename Type > constexpr DataSize magrathea::DataSize::tera (const Type & amount = Type (1)) [static]

Predefined data size in terabytes.

Constructs and returns a data size based on an amount of terabytes.

Template Parameters

Type	(Data type.)
------	--------------

Parameters

in	amount	Amount of terabytes.
----	--------	----------------------

Returns

Copy of the data size.

6.21.3.50 template<typename Type > constexpr Type magrathea::DataSize::terabytes () const

Get data size in terabytes.

Returns the data size in terabytes in the provided type.

Template Parameters

Type	Data type.
------	------------

Returns

The amount of terabytes.

6.21.3.51 template<typename Type > DataSize & magrathea::DataSize::terabytes (const Type & amount) [inline]

Set data size in terabytes.

Sets the current data size in terabytes.

Template Parameters

Type	(Data type.)
------	--------------

Parameters

in	<i>amount</i>	Amount of terabytes.
----	---------------	----------------------

Returns

Self reference.

6.21.3.52 constexpr bool magrathea::DataSize::valid () const

Get the validity of the data size.

Returns whether the data exists or not. The existence of data is defined as a size greater or equal to zero.

Returns

Data validity.

6.21.3.53 DataSize & magrathea::DataSize::valid (const bool *ok*) [inline]

Set valid status.

Sets the size to invalid if the argument is false, change it to the maximum between the current size and zero if true.

Parameters

in	<i>ok</i>	Status.
----	-----------	---------

Returns

Self reference.

6.21.4 Friends And Related Function Documentation**6.21.4.1 std::ostream& operator<< (std::ostream & *lhs*, const DataSize & *rhs*) [friend]**

[Output](#) stream operator.

Prints out the data size in the most convenient binary unit.

Parameters

in, out	<i>lhs</i>	Left-hand side stream.
in	<i>rhs</i>	Right-hand side data size.

Returns

[Output](#) stream.

6.21.5 Member Data Documentation**6.21.5.1 long long int magrathea::DataSize::_size [protected]**

Data size in bytes.

The documentation for this exception was generated from the following file:

- /data/home/mbreton/magrathea_pathfinder/src/magrathea/[datasize.h](#)

6.22 magrathea::EulerianCategory Exception Reference

Category concept of eulerian : data at a fixed position.

```
#include <euleriancategory.h>
```

Static Public Member Functions

Test

- static int [example \(\)](#)
Example function.

6.22.1 Detailed Description

Category concept of eulerian : data at a fixed position.

Categorizes data as an eulerian one. An eulerian data is specified at a fixed location : this is a way of looking at a fluid motion that focuses on specific locations in the space through which the fluid flows as time passes.

6.22.2 Member Function Documentation

6.22.2.1 int magrathea::EulerianCategory::example () [static]

Example function.

Tests and demonstrates the use of [EulerianCategory](#).

Returns

0 if no error.

The documentation for this exception was generated from the following file:

- /data/home/mbreton/magrathea_pathfinder/src/magrathea/[euleriancategory.h](#)

6.23 magrathea::Evolution< Type, Container > Exception Template Reference

Resizable container of steps dedicated to integration.

```
#include <evolution.h>
```

Public Member Functions

Lifecycle

- template<class OtherType >
[Evolution](#) (const std::initializer_list< OtherType > &source)
Implicit initializer list constructor.
- template<class OtherType , class OtherContainer >
[Evolution](#) (const [Evolution](#)< OtherType, OtherContainer > &source)
Explicit conversion constructor.

- template<class... Misc, class = typename std::enable_if<std::is_constructible<Container, Misc...>::value>::type> **Evolution** (Misc &&...misc)
Explicit generic constructor.

Operators

- template<class OtherType > **Evolution**< Type, Container > & **operator=** (const std::initializer_list< OtherType > &source)
Initializer list assignment operator.
- template<class OtherType , class OtherContainer > **Evolution**< Type, Container > & **operator=** (const **Evolution**< OtherType, OtherContainer > &source)
Conversion assignment operator.
- template<class Misc , class = typename std::enable_if<std::is_convertible<Container, typename std::remove_cv<typename std::remove_reference<Misc>::type>::type>::value>::type> **Evolution**< Type, Container > & **operator=** (Misc &&misc)
Generic assignment operator.
- bool **operator==** (const **Evolution**< Type, Container > &rhs) const
Equal to.
- bool **operator!=** (const **Evolution**< Type, Container > &rhs) const
Not equal to.
- bool **operator>** (const **Evolution**< Type, Container > &rhs) const
Greater than.
- bool **operator<** (const **Evolution**< Type, Container > &rhs) const
Less than.
- bool **operator>=** (const **Evolution**< Type, Container > &rhs) const
Greater than or equal to.
- bool **operator<=** (const **Evolution**< Type, Container > &rhs) const
Less than or equal to.
- Type & **operator[]** (const unsigned int i)
Direct access to the element.
- const Type & **operator[]** (const unsigned int i) const
Immutable direct access to the element.

Assignment

- **Evolution**< Type, Container > & **assign** (const **Evolution**< Type, Container > &source)
Copy assignment.
- template<class OtherType > **Evolution**< Type, Container > & **assign** (const std::initializer_list< OtherType > &source)
Initializer list assignment.
- template<class OtherType , class OtherContainer > **Evolution**< Type, Container > & **assign** (const **Evolution**< OtherType, OtherContainer > &source)
Conversion assignment.
- template<class... Misc, class = typename std::enable_if<std::is_constructible<Container, Misc...>::value>::type> **Evolution**< Type, Container > & **assign** (Misc &&...misc)
Generic assignment.

Management

- **Evolution**< Type, Container > & **nullify** ()
Nullify.
- **Evolution**< Type, Container > **copy** () const
Copy.
- template<class OtherType = Type, class OtherContainer = std::vector<OtherType>> **Evolution**< OtherType, OtherContainer > **cast** () const
Cast.

Access

- Type & `at` (const unsigned int i)
Access with range-check.
- const Type & `at` (const unsigned int i) const
Immutable access with range-check.
- Type & `front` (const unsigned int i=0)
Access to the i-th element from the beginning.
- const Type & `front` (const unsigned int i=0) const
Immutable access to the i-th element from the beginning.
- Type & `back` (const unsigned int i=0)
Access to the i-th element from the end.
- const Type & `back` (const unsigned int i=0) const
Immutable access to the i-th element from the end.
- Type & `cycle` (const int i)
Cyclic access to elements.
- const Type & `cycle` (const int i) const
Immutable cyclic access to elements.
- Container & `container` ()
Direct access to the underlying container.
- const Container & `container` () const
Direct access to the underlying container.
- Type * `data` ()
Direct access to the underlying array.
- const Type * `data` () const
Immutable direct access to the underlying array.

Iterators

- Type * `begin` ()
Iterator to the beginning.
- const Type * `begin` () const
Immutable iterator to the beginning.
- const Type * `cbegin` () const
Forced immutable iterator to the beginning.
- Type * `end` ()
Iterator to the end.
- const Type * `end` () const
Immutable iterator to the end.
- const Type * `cend` () const
Forced immutable iterator to the end.
- std::reverse_iterator< Type * > `rbegin` ()
Reverse iterator to the beginning.
- std::reverse_iterator< const Type * > `rbegin` () const
Immutable reverse iterator to the beginning.
- std::reverse_iterator< const Type * > `crbegin` () const
Forced immutable reverse iterator to the beginning.
- std::reverse_iterator< Type * > `rend` ()
Reverse iterator to the end.
- std::reverse_iterator< const Type * > `rend` () const
Immutable reverse iterator to the end.
- std::reverse_iterator< const Type * > `crend` () const
Forced immutable reverse iterator to the end.
- template<typename Iterator>
 unsigned int `index` (const Iterator &it) const

Index of an iterator in the container.

Capacity

- bool `empty` () const
Emptiness checking.
- unsigned int `size` () const
Number of elements.
- unsigned int `capacity` () const
Capacity of the underlying storage.
- `Evolution< Type, Container >` & `reserve` (const unsigned int n)
Storage reservation.
- `Evolution< Type, Container >` & `shrink` ()
Storage shrinking.
- unsigned long long int `space` () const
Available space.

Modifiers

- `Evolution< Type, Container >` & `fullclear` ()
Full clear the container.
- `Evolution< Type, Container >` & `clear` ()
Clear the contents.
- template<class... Misc>
`Evolution< Type, Container >` & `resize` (const Misc &...misc)
Resize.
- `Evolution< Type, Container >` & `pop` ()
Pop back.
- template<class OtherType , class = typename std::enable_if<std::is_convertible<OtherType, Type>::value>::type>
`Evolution< Type, Container >` & `append` (const OtherType &value)
Append an element.

Static Public Member Functions

Test

- static int `example` ()
Example function.

Protected Attributes

Data members

- Container `_container`
Internal container.

Friends

Stream

- template<class SelfType , class SelfContainer >
std::ostream & `operator<<` (std::ostream &lhs, const `Evolution< SelfType, SelfContainer >` &rhs)
Output stream operator.

6.23.1 Detailed Description

`template<class Type = Step<>, class Container = std::vector<Type>>exception magrathea::Evolution< Type, Container >`

Resizable container of steps dedicated to integration.

An evolution container is an accumulator of integration steps. It has the standard functions of containers plus additional functions to interpolate and sort the results.

Template Parameters

<i>Type</i>	Step type.
<i>Container</i>	Container type.

6.23.2 Constructor & Destructor Documentation

6.23.2.1 `template<class Type , class Container > template<class OtherType > magrathea::Evolution< Type, Container >::Evolution (const std::initializer_list< OtherType > & source) [inline]`

Implicit initializer list constructor.

Provides an implicit conversion from an initializer list.

Template Parameters

<i>OtherType</i>	(Other step type.)
------------------	--------------------

Parameters

<i>in</i>	<i>source</i>	Source of the copy.
-----------	---------------	---------------------

6.23.2.2 `template<class Type , class Container > template<class OtherType , class OtherContainer > magrathea::Evolution< Type, Container >::Evolution (const Evolution< OtherType, OtherContainer > & source) [inline], [explicit]`

Explicit conversion constructor.

Provides an explicit construction from another type of evolution container.

Template Parameters

<i>OtherType</i>	(Other step type.)
<i>OtherContainer</i>	(Other container type.)

Parameters

<i>in</i>	<i>source</i>	Source of the copy.
-----------	---------------	---------------------

6.23.2.3 `template<class Type , class Container > template<class... Misc, class > magrathea::Evolution< Type, Container >::Evolution (Misc &&... misc) [inline], [explicit]`

Explicit generic constructor.

Provides a generic interface to all constructors of the container.

Template Parameters

<i>Misc</i>	(Miscellaneous types.)
-------------	------------------------

Parameters

<i>in</i>	<i>misc</i>	Miscellaneous arguments.
-----------	-------------	--------------------------

6.23.3 Member Function Documentation

6.23.3.1 `template<class Type , class Container > template<class OtherType , class > Evolution< Type, Container > & magrathea::Evolution< Type, Container >::append (const OtherType & value) [inline]`

Append an element.

Appends an element to the end of the container.

Template Parameters

<i>OtherType</i>	(Other step type.)
------------------	--------------------

Parameters

<i>in</i>	<i>value</i>	Value of the element to be pushed back.
-----------	--------------	---

Returns

Self reference.

6.23.3.2 `template<class Type , class Container > Evolution< Type, Container > & magrathea::Evolution< Type, Container >::assign (const Evolution< Type, Container > & source) [inline]`

Copy assignment.

Provides a copy assignment from another evolution container.

Parameters

<i>in</i>	<i>source</i>	Source of the copy.
-----------	---------------	---------------------

Returns

Self reference.

6.23.3.3 `template<class Type , class Container > template<class OtherType > Evolution< Type, Container > & magrathea::Evolution< Type, Container >::assign (const std::initializer_list< OtherType > & source) [inline]`

Initializer list assignment.

Provides an assignment from an initializer list equivalent to a call to the constructor.

Template Parameters

<i>OtherType</i>	(Other step type.)
------------------	--------------------

Parameters

<code>in</code>	<code>source</code>	Source of the copy.
-----------------	---------------------	---------------------

Returns

Self reference.

6.23.3.4 template<class Type , class Container > template<class OtherType , class OtherContainer > Evolution< Type, Container > & magrathea::Evolution< Type, Container >::assign (const Evolution< OtherType, OtherContainer > & source) [inline]

Conversion assignment.

Provides an assignment from another type of evolution container.

Template Parameters

<code>OtherType</code>	(Other step type.)
<code>OtherContainer</code>	(Other container type.)

Parameters

<code>in</code>	<code>source</code>	Source of the copy.
-----------------	---------------------	---------------------

Returns

Self reference.

6.23.3.5 template<class Type , class Container > template<class... Misc, class > Evolution< Type, Container > & magrathea::Evolution< Type, Container >::assign (Misc &&... misc) [inline]

Generic assignment.

Provides a generic interface to all assignments of the container.

Template Parameters

<code>Misc</code>	(Miscellaneous types.)
-------------------	------------------------

Parameters

<code>in</code>	<code>misc</code>	Miscellaneous arguments.
-----------------	-------------------	--------------------------

Returns

Self reference.

6.23.3.6 template<class Type , class Container > Type & magrathea::Evolution< Type, Container >::at (const unsigned int i) [inline]

Access with range-check.

Provides an access to the element with a range-check.

Parameters

in	<i>i</i>	Index of the element.
----	----------	-----------------------

Returns

Reference to the element.

6.23.3.7 template<class Type , class Container > const Type & magrathea::Evolution< Type, Container >::at (const unsigned int *i*) const [inline]

Immutable access with range-check.

Provides an access to the element with a range-check.

Parameters

in	<i>i</i>	Index of the element.
----	----------	-----------------------

Returns

Immutable reference to the element.

6.23.3.8 template<class Type , class Container > Type & magrathea::Evolution< Type, Container >::back (const unsigned int *i* = 0) [inline]

Access to the *i*-th element from the end.

Returns a reference to the *i*-th last element in the container without doing any range check.

Parameters

in	<i>i</i>	Index of the element.
----	----------	-----------------------

Returns

Reference to the element.

6.23.3.9 template<class Type , class Container > const Type & magrathea::Evolution< Type, Container >::back (const unsigned int *i* = 0) const [inline]

Immutable access to the *i*-th element from the end.

Returns a reference to the *i*-th last element in the container without doing any range check.

Parameters

in	<i>i</i>	Index of the element.
----	----------	-----------------------

Returns

Immutable reference to the element.

**6.23.3.10 template<class Type , class Container > Type * magrathea::Evolution< Type, Container >::begin ()
[inline]**

Iterator to the beginning.

Returns a pointer to the first element.

Returns

Pointer to the beginning.

**6.23.3.11 template<class Type , class Container > const Type * magrathea::Evolution< Type, Container >::begin ()
const [inline]**

Immutable iterator to the beginning.

Returns a pointer to the first element.

Returns

Immutable pointer to the beginning.

**6.23.3.12 template<class Type , class Container > unsigned int magrathea::Evolution< Type, Container >::capacity ()
const [inline]**

Capacity of the underlying storage.

Returns the number of elements that the container has currently allocated space for.

Returns

Capacity of the currently allocated storage.

**6.23.3.13 template<class Type , class Container > template<class OtherType , class OtherContainer > Evolution<
OtherType, OtherContainer > magrathea::Evolution< Type, Container >::cast () const [inline]**

Cast.

Casts contents to another object type.

Template Parameters

<i>OtherType</i>	Other step type.
<i>OtherContainer</i>	Other container type.

Returns

Casted copy.

6.23.3.14 template<class Type , class Container > const Type * magrathea::Evolution< Type, Container >::cbegin() const [inline]

Forced immutable iterator to the beginning.

Returns a constant pointer to the first element.

Returns

Immutable pointer to the beginning.

6.23.3.15 template<class Type , class Container > const Type * magrathea::Evolution< Type, Container >::cend() const [inline]

Forced immutable iterator to the end.

Returns a constant pointer to the position after the last element.

Returns

Immutable pointer to the end.

6.23.3.16 template<class Type , class Container > Evolution< Type, Container > & magrathea::Evolution< Type, Container >::clear() [inline]

Clear the contents.

Removes all elements from the container.

Returns

Self reference.

6.23.3.17 template<class Type , class Container > Container & magrathea::Evolution< Type, Container >::container() [inline]

Direct access to the underlying container.

Provides a direct access to the underlying container by returning a reference to it.

Returns

Reference to the underlying container.

6.23.3.18 template<class Type , class Container > const Container & magrathea::Evolution< Type, Container >::container() const [inline]

Direct access to the underlying container.

Provides a direct access to the underlying container by returning a reference to it.

Returns

Immutable reference to the underlying container.

6.23.3.19 template<class Type , class Container > Evolution< Type, Container > **magrathea::Evolution< Type, Container >::copy() const** [inline]

Copy.

Generates a copy of the object.

Returns

Copy.

6.23.3.20 template<class Type , class Container > std::reverse_iterator< const Type * > **magrathea::Evolution< Type, Container >::crbegin() const** [inline]

Forced immutable reverse iterator to the beginning.

Returns a constant reversed pointer to the position after the last element.

Returns

Immutable pointer to the end.

6.23.3.21 template<class Type , class Container > std::reverse_iterator< const Type * > **magrathea::Evolution< Type, Container >::crend() const** [inline]

Forced immutable reverse iterator to the end.

Returns a constant reversed pointer to the first element.

Returns

Immutable pointer to the beginning.

6.23.3.22 template<class Type , class Container > Type & **magrathea::Evolution< Type, Container >::cycle(const int i)** [inline]

Cyclic access to elements.

Provides a cyclic access to the elements, using the index modulo. Negative indexes are supported. It allows to iterate several times over the contents just by incrementing the provided index.

Parameters

in	i	Index of the element.
----	---	-----------------------

Returns

Reference to the element.

6.23.3.23 template<class Type , class Container > const Type & **magrathea::Evolution< Type, Container >::cycle(const int i) const** [inline]

Immutable cyclic access to elements.

Provides a cyclic access to the elements, using the index modulo. Negative indexes are supported. It allows to iterate several times over the contents just by incrementing the provided index.

Parameters

in	i	Index of the element.
----	---	-----------------------

Returns

Immutable reference to the element.

6.23.3.24 template<class Type , class Container > Type * magrathea::Evolution< Type, Container >::data () [inline]

Direct access to the underlying array.

Provides a direct access to the underlying array by returning a pointer to the first element of storage.

Returns

Pointer to the underlying element storage.

6.23.3.25 template<class Type , class Container > const Type * magrathea::Evolution< Type, Container >::data () const [inline]

Immutable direct access to the underlying array.

Provides a direct access to the underlying array by returning a pointer to the first element of storage.

Returns

Immutable pointer to the underlying element storage.

6.23.3.26 template<class Type , class Container > bool magrathea::Evolution< Type, Container >::empty () const [inline]

Emptiness checking.

Checks if the container has no elements.

Returns

True if empty, false otherwise.

6.23.3.27 template<class Type , class Container > Type * magrathea::Evolution< Type, Container >::end () [inline]

Iterator to the end.

Returns a pointer to the position after the last element.

Returns

Pointer to the end.

```
6.23.3.28 template<class Type , class Container > const Type * magrathea::Evolution< Type, Container >::end ( )
const [inline]
```

Immutable iterator to the end.

Returns a pointer to the position after the last element.

Returns

Immutable pointer to the end.

```
6.23.3.29 template<class Type , class Container > int magrathea::Evolution< Type, Container >::example ( )
[static]
```

Example function.

Tests and demonstrates the use of [Evolution](#).

Returns

0 if no error.

```
6.23.3.30 template<class Type , class Container > Type & magrathea::Evolution< Type, Container >::front ( const
unsigned int i = 0 ) [inline]
```

Access to the i-th element from the beginning.

Returns a reference to the i-th first element in the container without doing any range check.

Parameters

in	i	Index of the element.
----	---	-----------------------

Returns

Reference to the element.

```
6.23.3.31 template<class Type , class Container > const Type & magrathea::Evolution< Type, Container >::front ( const
unsigned int i = 0 ) const [inline]
```

Immutable access to the i-th element from the beginning.

Returns a reference to the i-th first element in the container without doing any range check.

Parameters

in	i	Index of the element.
----	---	-----------------------

Returns

Immutable reference to the element.

```
6.23.3.32 template<class Type , class Container > Evolution< Type, Container > & magrathea::Evolution< Type,
Container >::fullclear ( ) [inline]
```

Full clear the container.

Erase the container.

Returns

Self reference.

6.23.3.33 template<class Type , class Container > template<typename Iterator > unsigned int magrathea::Evolution< Type, Container >::index (const Iterator & *it*) const [inline]

Index of an iterator in the container.

Returns the index of the element pointed by an iterator or a pointer.

Template Parameters

<i>Iterator</i>	(Pointer or iterator type.)
-----------------	-----------------------------

Parameters

in	<i>it</i>	Iterator to the element.
----	-----------	--------------------------

Returns

Index of the element.

Exceptions

<i>std::out_of_range</i>	Out of range.
--------------------------	---------------

6.23.3.34 template<class Type , class Container > Evolution< Type, Container > & magrathea::Evolution< Type, Container >::nullify () [inline]

Nullify.

Resets all data members to their default values.

Returns

Self reference.

6.23.3.35 template<class Type , class Container > bool magrathea::Evolution< Type, Container >::operator!= (const Evolution< Type, Container > & *rhs*) const [inline]

Not equal to.

Lexicographically compares the values in the container for difference.

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

True if not equal, false if equal.

6.23.3.36 template<class Type , class Container > bool **magrathea::Evolution< Type, Container >::operator<(const Evolution< Type, Container > & rhs) const** [inline]

Less than.

Lexicographically compares the values in the container for the less than comparison operator.

Parameters

in	rhs	Right-hand side.
----	-----	------------------

Returns

True if less than, false otherwise.

6.23.3.37 template<class Type , class Container > bool **magrathea::Evolution< Type, Container >::operator<=(const Evolution< Type, Container > & rhs) const** [inline]

Less than or equal to.

Lexicographically compares the values in the container for the less than or equal to comparison operator.

Parameters

in	rhs	Right-hand side.
----	-----	------------------

Returns

True if less than or equal to, false otherwise.

6.23.3.38 template<class Type , class Container > template<class OtherType > **Evolution< Type, Container > & magrathea::Evolution< Type, Container >::operator=(const std::initializer_list< OtherType > & source)** [inline]

Initializer list assignment operator.

Provides an initializer list assignment.

Template Parameters

OtherType	(Other step type.)
-----------	--------------------

Parameters

in	source	Source of the copy.
----	--------	---------------------

Returns

Self reference.

6.23.3.39 template<class Type , class Container > template<class OtherType , class OtherContainer > **Evolution< Type, Container > & magrathea::Evolution< Type, Container >::operator=(const Evolution< OtherType, OtherContainer > & source)** [inline]

Conversion assignment operator.

Provides an assignment from another type of evolution container.

Template Parameters

<i>OtherType</i>	(Other step type.)
<i>OtherContainer</i>	(Other container type.)

Parameters

in	<i>source</i>	Source of the copy.
----	---------------	---------------------

Returns

Self reference.

6.23.3.40 template<class Type , class Container > template<class Misc , class > Evolution< Type, Container > & magrathea::Evolution< Type, Container >::operator=(Misc && misc) [inline]

Generic assignment operator.

Provides a generic interface to all assignment operators of the container

Template Parameters

<i>Misc</i>	(Miscellaneous type.)
-------------	-----------------------

Parameters

in	<i>misc</i>	Miscellaneous argument.
----	-------------	-------------------------

Returns

Self reference.

6.23.3.41 template<class Type , class Container > bool magrathea::Evolution< Type, Container >::operator==(const Evolution< Type, Container > & rhs) const [inline]

Equal to.

Lexicographically compares the values in the container for equality.

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

True if equal, false if not equal.

6.23.3.42 template<class Type , class Container > bool magrathea::Evolution< Type, Container >::operator> (const Evolution< Type, Container > & rhs) const [inline]

Greater than.

Lexicographically compares the values in the container for the greater than comparison operator.

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

True if greater than, false otherwise.

6.23.3.43 template<class Type , class Container > bool magrathea::Evolution< Type, Container >::operator>=(const Evolution< Type, Container > & *rhs*) const [inline]

Greater than or equal to.

Lexicographically compares the values in the container for the greater than or equal to comparison operator.

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

True if greater than or equal to, false otherwise.

6.23.3.44 template<class Type , class Container > Type & magrathea::Evolution< Type, Container >::operator[](const unsigned int *i*) [inline]

Direct access to the element.

Provides a direct access to the specified element.

Parameters

in	<i>i</i>	Index of the element.
----	----------	-----------------------

Returns

Reference to the element.

6.23.3.45 template<class Type , class Container > const Type & magrathea::Evolution< Type, Container >::operator[](const unsigned int *i*) const [inline]

Immutable direct access to the element.

Provides a direct access to the specified element.

Parameters

in	<i>i</i>	Index of the element.
----	----------	-----------------------

Returns

Immutable reference to the element.

6.23.3.46 template<class Type , class Container > Evolution< Type, Container > & magrathea::Evolution< Type, Container >::pop() [inline]

Pop back.

Removes the last element of the container.

Returns

Self reference.

6.23.3.47 template<class Type , class Container > std::reverse_iterator< Type * > magrathea::Evolution< Type, Container >::rbegin() [inline]

Reverse iterator to the beginning.

Returns a reversed pointer to the position after the last element.

Returns

Pointer to the end.

6.23.3.48 template<class Type , class Container > std::reverse_iterator< const Type * > magrathea::Evolution< Type, Container >::rbegin() const [inline]

Immutable reverse iterator to the beginning.

Returns a reversed pointer to the position after the last element.

Returns

Immutable pointer to the end.

6.23.3.49 template<class Type , class Container > std::reverse_iterator< Type * > magrathea::Evolution< Type, Container >::rend() [inline]

Reverse iterator to the end.

Returns a reversed pointer to the first element.

Returns

Pointer to the beginning.

6.23.3.50 template<class Type , class Container > std::reverse_iterator< const Type * > magrathea::Evolution< Type, Container >::rend() const [inline]

Immutable reverse iterator to the end.

Returns a reversed pointer to the first element.

Returns

Immutable pointer to the beginning.

6.23.3.51 template<class Type , class Container > Evolution< Type, Container > & magrathea::Evolution< Type, Container >::reserve (const unsigned int *n*) [inline]

Storage reservation.

Increases the capacity of the underlying storage. Existing elements are protected so it could not invalidate the actual contents.

Parameters

in	<i>n</i>	New size.
----	----------	-----------

Returns

Self reference.

6.23.3.52 template<class Type , class Container > template<class... Misc> Evolution< Type, Container > & magrathea::Evolution< Type, Container >::resize (const Misc &... *misc*) [inline]

Resize.

Resizes the container to contain a new number of elements.

Template Parameters

Misc	(Miscellaneous types.)
------	------------------------

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Self reference.

6.23.3.53 template<class Type , class Container > Evolution< Type, Container > & magrathea::Evolution< Type, Container >::shrink () [inline]

Storage shrinking.

Reduces memory usage by freeing unused memory.

Returns

Self reference.

6.23.3.54 template<class Type , class Container > unsigned int magrathea::Evolution< Type, Container >::size () const [inline]

Number of elements.

Returns the distance between the first and the last element.

Returns

The number of elements in the container.

6.23.3.55 template<class Type , class Container > unsigned long long int magrathea::Evolution< Type, Container >::space() const [inline]

Available space.

Returns the maximum possible number of elements.

Returns

Maximum number of elements.

6.23.4 Friends And Related Function Documentation

6.23.4.1 template<class Type = Step<>, class Container = std::vector<Type>> template<class SelfType , class SelfContainer > std::ostream& operator<< (std::ostream & lhs, const Evolution< SelfType, SelfContainer > & rhs) [friend]

[Output](#) stream operator.

Adds each element to the stream using the `fill()` character as a separator.

Template Parameters

<i>SelfType</i>	(Step type.)
<i>SelfContainer</i>	(Container type.)

Parameters

<i>in,out</i>	<i>lhs</i>	Left-hand side stream.
<i>in</i>	<i>rhs</i>	Right-hand side object.

Returns

[Output](#) stream.

6.23.5 Member Data Documentation

6.23.5.1 template<class Type = Step<>, class Container = std::vector<Type>> Container magrathea::Evolution< Type, Container >::container [protected]

Internal container.

The documentation for this exception was generated from the following file:

- /data/home/mbreton/magrathea_pathfinder/src/magrathea/[evolution.h](#)

6.24 magrathea::FileList Exception Reference

List of files based on a function or a vector.

```
#include <filelist.h>
```

Public Member Functions

Lifecycle

- template<class Type = std::string, class = typename std::enable_if<!std::is_convertible<Type, std::string>::value>::type>
FileList (const std::initializer_list< Type > &source=std::initializer_list< Type >())
Implicit initializer list constructor.
- template<class ContainerType , class PrefixType = std::string, class = typename std::enable_if<!std::is_convertible<ContainerType, std::string>::value> && (std::is_convertible<typename std::remove_reference<decltype(*std::begin(std::declval<ContainerType>()))>::type, std::string>::value) && (std::is_convertible<PrefixType, std::string>::value)>::type>
FileList (const ContainerType &source, const PrefixType &prefix=PrefixType())
Explicit container constructor.
- template<typename IteratorType , class PrefixType = std::string, class = typename std::enable_if<(std::is_convertible<typename std::remove_reference<decltype(*std::declval<IteratorType>())>::type, std::string>::value) && (std::is_convertible<PrefixType, std::string>::value)>::type>
FileList (const IteratorType &begin, const IteratorType &end, const PrefixType &prefix=PrefixType())
Explicit iterator constructor.
- template<class FormatType , class PrefixType = std::string, class = typename std::enable_if<(std::is_convertible<FormatType, std::string>::value) && (std::is_convertible<PrefixType, std::string>::value)>::type>
FileList (const FormatType &cformat, const int width=0, const unsigned int amount=1, const unsigned int shift=0, const PrefixType &prefix=PrefixType())
Explicit format constructor.
- template<class Function , class PrefixType = std::string, class = typename std::enable_if<(std::is_convertible<typename std::remove_reference<typename std::result_of<Function(const unsigned int)>::type>::type, std::string>::value) && !(std::is_same<typename std::decay<Function>::type, FileList>::value) && (std::is_convertible<PrefixType, std::string>::value)>::type>
FileList (Function &&f, const unsigned int amount=1, const unsigned int shift=0, const PrefixType &prefix=PrefixType())
Explicit generator constructor.

Operators

- template<class ContainerType , class = typename std::enable_if<!std::is_convertible<ContainerType, std::string>::value> && (std::is_convertible<typename std::remove_reference<decltype(*std::begin(std::declval<ContainerType>()))>::type, std::string>::value)>::type>
FileList & operator= (const ContainerType &source)
Container assignment operator.
- template<class FormatType , class = typename std::enable_if<std::is_convertible<FormatType, std::string>::value>::type, class = void>
FileList & operator= (const FormatType &cformat)
Format assignment operator.
- template<class Function , class = typename std::enable_if<(std::is_convertible<typename std::remove_reference<typename std::result_of<Function(const unsigned int)>::type>::type, std::string>::value) && !(std::is_same<typename std::decay<Function>::type, FileList>::value)>::type, class = void, class = void>
FileList & operator= (Function &&f)
Generator assignment operator.
- bool **operator==** (const **FileList** &rhs) const
Equal to.
- bool **operator!=** (const **FileList** &rhs) const
Not equal to.

Assignment

- **FileList & assign** (const **FileList** &source)
Copy assignment.
- template<class ContainerType = std::initializer_list<std::string>, class PrefixType = std::string, class = typename std::enable_if<!std::is_convertible<ContainerType, std::string>::value> && (std::is_convertible<typename std::remove_reference<decltype(*std::begin(std::declval<ContainerType>()))>::type, std::string>::value) && (std::is_convertible<PrefixType, std::string>::value)>::type>
FileList & assign (const ContainerType &source=ContainerType(), const PrefixType &prefix=PrefixType())
Container assignment.

- template<typename IteratorType , class PrefixType = std::string, class = typename std::enable_if<(std::is_convertible<typename std::remove_reference<decltype(*std::declval<IteratorType>())>::type, std::string>::value) && (std::is_convertible<PrefixType, std::string>::value)>::type>
 FileList & assign (const IteratorType &begin, const IteratorType &end, const PrefixType &prefix=PrefixType())
 Iterator assignment.
- template<class FormatType , class PrefixType = std::string, class = typename std::enable_if<(std::is_convertible<FormatType, std::string>::value) && (std::is_convertible<PrefixType, std::string>::value)>::type>
 FileList & assign (const FormatType &cformat, const int width=0, const unsigned int amount=1, const unsigned int shift=0, const PrefixType &prefix=PrefixType())
 Format assignment.
- template<class Function , class PrefixType = std::string, class = typename std::enable_if<(std::is_convertible<typename std::remove_reference<typename std::result_of<Function(const unsigned int)>::type, std::string>::value) && !(std::is_same<typename std::decay<Function>::type, FileList>::value) && (std::is_convertible<PrefixType, std::string>::value)>::type>
 FileList & assign (Function &&f, const unsigned int amount=1, const unsigned int shift=0, const PrefixType &prefix=PrefixType())
 Generator assignment.

Management

- bool **empty** () const
 Empty.
- bool **cleared** () const
 Cleared.
- unsigned int **capacity** () const
 Capacity.
- **FileList & resize** (const unsigned int amount)
 Resize.
- **FileList & reserve** (const unsigned int amount)
 Reserve space for the container.
- **FileList & shrink** ()
 Shrink the container.
- **FileList & clear** ()
 Clear.
- **FileList copy** () const
 Copy.
- template<class Type = FileList, class = typename std::enable_if<std::is_constructible<Type, FileList>::value>::type>
 Type cast () const
 Cast.

Getters

- const std::vector< std::string > & **container** () const
 Container getter.
- const std::string & **format** () const
 Format getter.
- const std::function< std::string(const unsigned int)> & **generator** () const
 Generator getter.
- const std::string & **root** () const
 Root getter.
- const int & **length** () const
 Length getter.
- const unsigned int & **size** () const
 Size getter.
- const unsigned int & **offset** () const
 Offset getter.

Files

- std::string **operator[]** (const unsigned int i) const
Subscript operator.
- std::string **at** (const unsigned int i) const
Range check access.
- std::string **front** (const unsigned int i=0) const
Access from the beginning.
- std::string **back** (const unsigned int i=0) const
Access from the end.
- std::string **operator()** (const unsigned int i) const
Generator operator.
- std::string **get** (const unsigned int i) const
Generator getter.

Manipulations

- template<class Type = std::vector<std::string>, class = typename std::enable_if<((std::is_void<decltype(std::declval<Type>().
resize(0))>::value) || (!std::is_void<decltype(std::declval<Type>().resize(0))>::value)) && (!std::is_const<decltype(std::declval<
Type>()[0])>::value) && (std::is_reference<decltype(std::declval<Type>()[0])>::value) && (std::is_convertible<decltype(std-
::declval<Type>()[0]), std::string>::value)>::type>
Type convert () const
Convert.
- std::string **common** () const
Common file part.
- **FileList formatify** () const
Detect common format.

Algorithms

- template<class Function , class... Args, class = typename std::result_of<Function(std::string, Args...)>::type>
FileList & apply (Function &&f, Args &&...args)
Application of a function on each file.
- unsigned int **count** () const
Count existing files.
- template<class Function , class = typename std::enable_if<std::is_convertible<typename std::result_of<Function(std::string)>-
::type, bool>::value>::type>
unsigned int **count** (Function &&f) const
Count files satisfying a condition.
- template<class Function , typename FirstType = std::true_type, typename AmountType = std::true_type, class = typename std-
::enable_if<((std::is_convertible<typename std::result_of<Function(std::string)>::type, bool)>::value) && ((std::is_same<Amount-
Type, std::true_type>::value) || (std::is_convertible<FirstType, unsigned int>::value)) && ((std::is_same<AmountType, std::true_-
type>::value) || (std::is_convertible<AmountType, int>::value))>::type>
unsigned int **find** (Function &&f, const FirstType first=FirstType(), const AmountType amount=Amount-
Type()) const
Find first file satisfying a condition.
- template<class Function = std::less<std::string>, class = typename std::enable_if<std::is_convertible<typename std::result_of<-
Function(std::string, std::string)>::type, bool>::value>::type>
FileList & sort (Function &&f=Function())
Sort file names.
- template<class Function = std::less<std::string>, class = typename std::enable_if<std::is_convertible<typename std::result_of<-
Function(std::string, std::string)>::type, bool>::value>::type>
bool **sorted** (Function &&f=Function()) const
Check sorting.
- template<class Function = std::equal_to<std::string>, class = typename std::enable_if<std::is_convertible<typename std::result_ -
of<Function(std::string, std::string)>::type, bool>::value>::type>
FileList & unique (Function &&f=Function())
Unique file names.
- template<class Function = std::equal_to<std::string>, class = typename std::enable_if<std::is_convertible<typename std::result_ -
of<Function(std::string, std::string)>::type, bool>::value>::type>
bool **unicity** (Function &&f=Function()) const
Check unicity.

Static Public Member Functions

Helpers

- template<typename IndexType = std::true_type, class = typename std::enable_if<(std::is_same<IndexType, std::true_type>::value) || (std::is_convertible<IndexType, unsigned int>::value)>::type>
static std::string **generate** (const IndexType i=IndexType())
Temporary file name generator.
- template<class ContainerType , class = typename std::enable_if<!std::is_convertible<ContainerType, std::string>::value) && (std::is_convertible<typename std::remove_reference<decaytype(*std::begin(std::declval<ContainerType>()))>::type, std::string>::value)>::type>
static std::string **generate** (const unsigned int i, const ContainerType &source)
Container-based file name generator.
- template<unsigned int BufferSize = std::numeric_limits<unsigned char>::max()*std::numeric_limits<unsigned char>::digits, typename WidthType = std::true_type, class = typename std::enable_if<(std::is_same<WidthType, std::true_type>::value) || (std::is_convertible<WidthType, int>::value)>::type>
static std::string **generate** (const unsigned int i, const std::string &format, const WidthType width=WidthType())
Format-based file name generator.
- template<class Function , class = typename std::enable_if<(std::is_convertible<typename std::remove_reference<typename std::result_of<Function(const unsigned int)>::type, std::string>::value) && !(std::is_same<typename std::decay<Function>::type, FileList>::value)>::type>
static std::string **generate** (const unsigned int i, Function &&f)
Format-based file name generator.
- template<typename Dir = unsigned int, typename Min = unsigned int, typename Num = unsigned int, typename Max = unsigned int, class = typename std::enable_if<(std::is_convertible<Dir, unsigned int>::value) && (std::is_convertible<Min, unsigned int>::value) && (std::is_convertible<Num, unsigned int>::value) && (std::is_convertible<Max, unsigned int>::value)>::type>
static unsigned int **numberify** (const std::string &str, unsigned int pos=std::numeric_limits< unsigned int >::max(), Dir &&dir=Dir(), Min &&min=Min(), Num &&num=Num(), Max &&max=Max())
Parse number in a string.

Test

- static int **example** ()
Example function.

Public Attributes

Data members

- std::vector< std::string > **_container**
Container of file names.
- std::string **_format**
Format of file names.
- std::function< std::string(const unsigned int)> **_generator**
Generator of file names.
- std::string **_root**
Common root prefix.
- int **_length**
Optional width of the format field.
- unsigned int **_size**
Total file list size.
- unsigned int **_offset**
Offset between file identificator and list index.

Friends

Stream

- std::ostream & operator<< (std::ostream &lhs, const **FileList** &rhs)
Output stream operator.

6.24.1 Detailed Description

List of files based on a function or a vector.

Class to hold a list of paths to files. Internally, the list can be stored as a container of strings, a C-like format or a function object that will generate the file names on the fly. An offset allows to shift the real identifier of the files regarding the list index.

6.24.2 Constructor & Destructor Documentation

6.24.2.1 template<class Type , class > magrathea::FileList::FileList (const std::initializer_list< Type > & source = std:: initializer_list<Type>()) [inline]

Implicit constructor.

Implicitly constructs the file list from an initializer list of paths.

Template Parameters

Type	(Data type.)
------	--------------

Parameters

in	source	Source of the copy.
----	--------	---------------------

6.24.2.2 template<class ContainerType , class PrefixType , class > magrathea::FileList::FileList (const ContainerType & source, const PrefixType & prefix = PrefixType()) [inline], [explicit]

Explicit container constructor.

Explicitly constructs the file list from a container of paths.

Template Parameters

ContainerType	(Container type.)
PrefixType	(Prefix type.)

Parameters

in	source	Source of the copy.
in	prefix	Common root prefix.

6.24.2.3 template<typename IteratorType , class PrefixType , class > magrathea::FileList::FileList (const IteratorType & begin, const IteratorType & end, const PrefixType & prefix = PrefixType()) [inline], [explicit]

Explicit iterator constructor.

Explicitly constructs the file list from a range of iterators.

Template Parameters

<i>IteratorType</i>	(Iterator or pointer type.)
<i>PrefixType</i>	(Prefix type.)

Parameters

in	<i>begin</i>	Iterator to the beginning of the range.
in	<i>end</i>	Iterator to the end of the range.
in	<i>prefix</i>	Common root prefix.

```
6.24.2.4 template<class FormatType , class PrefixType , class > magrathea::FileList::FileList ( const FormatType & cformat,  
const int width = 0, const unsigned int amount = 1, const unsigned int shift = 0, const PrefixType & prefix =  
PrefixType() ) [inline], [explicit]
```

Explicit format constructor.

Explicitely constructs the file list from a C-like format string and its width if required.

Template Parameters

<i>FormatType</i>	(Type of the format string.)
<i>PrefixType</i>	(Prefix type.)

Parameters

in	<i>cformat</i>	Format of file names.
in	<i>width</i>	Optional width of the format field.
in	<i>amount</i>	Total file list size.
in	<i>shift</i>	Offset between file identificator and list index.
in	<i>prefix</i>	Common root prefix.

```
6.24.2.5 template<class Function , class PrefixType , class > magrathea::FileList::FileList ( Function && f, const unsigned  
int amount = 1, const unsigned int shift = 0, const PrefixType & prefix = PrefixType() ) [inline],  
[explicit]
```

Explicit generator constructor.

Explicitely constructs the file list from a generator function.

Template Parameters

<i>Function</i>	(Function type : std::string(const unsigned int).)
<i>PrefixType</i>	(Prefix type.)

Parameters

in	<i>f</i>	Function object std::string(const unsigned int).
in	<i>amount</i>	Total file list size.
in	<i>shift</i>	Offset between file identificator and list index.
in	<i>prefix</i>	Common root prefix.

6.24.3 Member Function Documentation

6.24.3.1 template<class Function , class... Args, class > **FileList & magrathea::FileList::apply (Function && f, Args &&... args)**

Application of a function on each file.

Applies the provided algorithm on each file of the list, passing the extra arguments as extra function parameters if specified.

Template Parameters

<i>Function</i>	(Function type : Type (std::string, Args...)) .)
<i>Args</i>	(Extra types.)

Parameters

in	<i>f</i>	Function object Type (std::string, Args...).
in	<i>args</i>	Extra arguments of the function.

Returns

Self reference.

6.24.3.2 **FileList & magrathea::FileList::assign (const FileList & source) [inline]**

Copy assignment.

Assigns the file list contents using another file list.

Parameters

in	<i>source</i>	Source of the copy.
----	---------------	---------------------

Returns

Self reference.

6.24.3.3 template<class ContainerType , class PrefixType , class > **FileList & magrathea::FileList::assign (const ContainerType & source = ContainerType (), const PrefixType & prefix = PrefixType ()) [inline]**

Container assignment.

Assigns the file list contents using a container of paths.

Template Parameters

<i>ContainerType</i>	(Container type.)
<i>PrefixType</i>	(Prefix type.)

Parameters

in	<i>source</i>	Source of the copy.
in	<i>prefix</i>	Common root prefix.

Returns

Self reference.

6.24.3.4 template<typename IteratorType , class PrefixType , class > **FileList** & magrathea::FileList::assign (const IteratorType & *begin*, const IteratorType & *end*, const PrefixType & *prefix* = PrefixType()) [inline]

Iterator assignment.

Assigns the file list contents using a range of iterators.

Template Parameters

<i>IteratorType</i>	(Iterator or pointer type.)
<i>PrefixType</i>	(Prefix type.)

Parameters

in	<i>begin</i>	Iterator to the beginning of the range.
in	<i>end</i>	Iterator to the end of the range.
in	<i>prefix</i>	Common root prefix.

Returns

Self reference.

6.24.3.5 template<class FormatType , class PrefixType , class > **FileList** & magrathea::FileList::assign (const FormatType & *cformat*, const int *width* = 0, const unsigned int *amount* = 1, const unsigned int *shift* = 0, const PrefixType & *prefix* = PrefixType()) [inline]

Format assignment.

Assigns the file list contents using a C-like format string and its width if required.

Template Parameters

<i>FormatType</i>	(Type of the format string.)
<i>PrefixType</i>	(Prefix type.)

Parameters

in	<i>cformat</i>	Format of file names.
in	<i>width</i>	Optional width of the format field.
in	<i>amount</i>	Total file list size.
in	<i>shift</i>	Offset between file identifier and list index.
in	<i>prefix</i>	Common root prefix.

Returns

Self reference.

6.24.3.6 template<class Function , class PrefixType , class > **FileList** & magrathea::FileList::assign (Function && *f*, const unsigned int *amount* = 1, const unsigned int *shift* = 0, const PrefixType & *prefix* = PrefixType()) [inline]

Generator assignment.

Assigns the file list contents usign a generator function.

Template Parameters

<i>Function</i>	(Function type: <code>std::string(const unsigned int.)</code>)
<i>PrefixType</i>	(Prefix type.)

Parameters

in	<i>f</i>	Function object <code>std::string(const unsigned int.)</code> .
in	<i>amount</i>	Total file list size.
in	<i>shift</i>	Offset between file identificator and list index.
in	<i>prefix</i>	Common root prefix.

Returns

Self reference.

6.24.3.7 `std::string magrathea::FileList::at(const unsigned int i) const [inline]`

Range check access.

Returns the file name at the specified position in the file list or throws an error if the index cannot be reached.

Parameters

in	<i>i</i>	Index.
----	----------	--------

Returns

Copy of the specified element.

Exceptions

<code>std::out_of_range</code>	Out of range.
--------------------------------	---------------

6.24.3.8 `std::string magrathea::FileList::back(const unsigned int i = 0) const [inline]`

Access from the end.

Returns the file name at the specified position in the file list starting from the end and without any range check.

Parameters

in	<i>i</i>	Index.
----	----------	--------

Returns

Copy of the specified element.

6.24.3.9 `unsigned int magrathea::FileList::capacity() const [inline]`

Capacity.

Returns the current capacity of the underlying container if it is used to store the file names, or returns the size if a format or a generator function is used.

Returns

Current capacity.

6.24.3.10 template<class Type , class > Type magrathea::FileList::cast() const [inline]

Cast.

Returns a copy of the file list casted to the provided type.

Template Parameters

Type	Type.
------	-------

Returns

Casted copy.

6.24.3.11 FileList & magrathea::FileList::clear() [inline]

Clear.

Clears the whole contents of the file list.

Returns

Self reference.

6.24.3.12 bool magrathea::FileList::cleared() const [inline]

Cleared.

Returns true if the file list is equal to a cleared one. It corresponds to an empty size, an empty format and an empty function.

Returns

True if the file list is cleared, false otherwise.

6.24.3.13 std::string magrathea::FileList::common() const

Common file part.

Detects the longest common character sequence within the whole list, starting from the beginning.

Returns

Longest common part.

6.24.3.14 const std::vector< std::string > & magrathea::FileList::container() const [inline]

Container getter.

Gets the container of file names.

Returns

Immutable reference to the container.

6.24.3.15 template<class Type , class > Type magrathea::FileList::convert() const

Convert.

Converts data to the specified type of container.

Template Parameters

Type	Type.
------	-------

Returns

Container of file names.

6.24.3.16 FileList magrathea::FileList::copy() const [inline]

Copy.

Returns a copy of the file list.

Returns

Copy.

6.24.3.17 unsigned int magrathea::FileList::count() const

Count existing files.

Counts the number of existing files in the list.

Returns

Number of existing files.

6.24.3.18 template<class Function , class > unsigned int magrathea::FileList::count(Function && f) const

Count files satisfying a condition.

Counts the number of files in the list that satisfy the provided condition.

Template Parameters

Function	(Function type : bool (std::string).)
----------	---------------------------------------

Parameters

in	f	Function object bool (std::string).
----	---	-------------------------------------

Returns

Number of files satisfying the condition.

6.24.3.19 bool magrathea::FileList::empty() const [inline]

Empty.

Returns true if the file list size is equal to zero.

Returns

True if the file list is empty, false otherwise.

6.24.3.20 int magrathea::FileList::example() [static]

Example function.

Tests and demonstrates the use of [FileList](#).

Returns

0 if no error.

6.24.3.21 template<class Function , typename FirstType , typename AmountType , class > unsigned int magrathea::FileList::find(Function && f, const FirstType first = FirstType(), const AmountType amount = AmountType()) const

Find first file satisfying a condition.

Computes the index of the first file satisfying the specified predicate. The first and amount arguments allows to specify the first file to test and the amount of file to test. As an example, an amount of zero file do not test any file, and an amount of -2 file tests the first file and the previous one. If no file satisfy the predicate, the function returns the size of the list. Boundary overflows and unsigned substractions are correctly handled.

Template Parameters

<i>Function</i>	(Function type: <code>bool (std::string::)</code>)
<i>FirstType</i>	(Integral type of the first index.)
<i>AmountType</i>	(Integral type of the first index.)

Parameters

in	<i>f</i>	Function object <code>bool (std::string::)</code> .
in	<i>first</i>	First file index to test.
in	<i>amount</i>	Amount of file to test in the corresponding direction.

Returns

Index of the first file satisfying the predicate.

6.24.3.22 const std::string & magrathea::FileList::format() const [inline]

Format getter.

Gets the format of file names.

Returns

Immutable reference to the format.

6.24.3.23 FileList magrathea::FileList::formatify() const

Detect common format.

Computes whether a common format can be deduced from the file list and returns a format based file list on success or an empty one on failure. Note that in case of a function or a container based list with a single element, the last number of the string is converted to a format.

Returns

Formatted file list.

6.24.3.24 std::string magrathea::FileList::front (const unsigned int *i* = 0) const [inline]

Access from the beginning.

Returns the file name at the specified position in the file list starting from the beginning and without any range check.

Parameters

in	<i>i</i>	Index.
----	----------	--------

Returns

Copy of the specified element.

6.24.3.25 template<typename IndexType , class > std::string magrathea::FileList::generate (const IndexType *i* = IndexType()) [inline], [static]

Temporary file name generator.

Generates a temporary file name based on the specified index.

Template Parameters

IndexType	(Integral type corresponding to the index.)
-----------	---

Parameters

in	<i>i</i>	Index.
----	----------	--------

Returns

File name.

6.24.3.26 template<class ContainerType , class > std::string magrathea::FileList::generate (const unsigned int *i*, const ContainerType & *source*) [inline], [static]

Container-based file name generator.

Gets the specified element of the passed container.

Template Parameters

ContainerType	(Container type.)
---------------	-------------------

Parameters

in	<i>i</i>	Index.
in	<i>source</i>	Source container.

Returns

File name.

```
6.24.3.27 template<unsigned int BufferSize, typename WidthType , class > std::string magrathea::FileList::generate (
    const unsigned int i, const std::string & cformat, const WidthType width = WidthType() ) [inline],
[static]
```

Format-based file name generator.

Generates a file name using the specified C-style format. If this format requires a width specifier, it can be passed as an optional parameter.

Template Parameters

<i>BufferSize</i>	Value of the internal buffer size.
<i>WidthType</i>	(Integral type corresponding to the format width.)

Parameters

in	<i>i</i>	Index.
in	<i>cformat</i>	C-like format string.
in	<i>width</i>	Optional width of the format field.

Returns

File name.

```
6.24.3.28 template<class Function , class > std::string magrathea::FileList::generate ( const unsigned int i, Function && f )
[inline], [static]
```

Format-based file name generator.

Generates a file name passing the index to the specified function.

Template Parameters

<i>Function</i>	(Function type : std::string (const unsigned int).)
-----------------	---

Parameters

in	<i>i</i>	Index.
in	<i>f</i>	Function object std::string (const unsigned int).

Returns

File name.

```
6.24.3.29 const std::function< std::string(const unsigned int)> & magrathea::FileList::generator ( ) const [inline]
```

Generator getter.

Gets the generator of file names.

Returns

Immutable reference to the generator.

6.24.3.30 std::string magrathea::FileList::get(const unsigned int *i*) const [inline]

Generator getter.

Generates the file name using the specified index and ignoring the root, the size limit (except if the list is based on a container) and the offset.

Parameters

in	<i>i</i>	Index.
----	----------	--------

Returns

Copy of the generated element.

6.24.3.31 const int & magrathea::FileList::length() const [inline]

Length getter.

Gets the optional width of the format field.

Returns

Immutable reference to the length.

6.24.3.32 template<typename Dir , typename Min , typename Num , typename Max , class > unsigned int magrathea::FileList::numberify(const std::string & str, unsigned int pos = std::numeric_limits<unsigned int>::max(), Dir && dir = Dir(), Min && min = Min(), Num && num = Num(), Max && max = Max()) [static]

Parse number in a string.

Detects positions of specific markers around a digit in a string. If the specified position is greater than the string length, it is first adjusted to the last character of the string. If the character at the provided position is not a digit, the function starts by searching the last digit before the position. In case of failure, all the markers are set equal to the end of the string.

Template Parameters

<i>Dir</i>	(Directory type.)
<i>Min</i>	(Minimum type.)
<i>Num</i>	(Number type.)
<i>Max</i>	(Maximum type.)

Parameters

in	<i>str</i>	Input string containing the number to parse.
in	<i>pos</i>	Position of a digit inside the string.
out	<i>dir</i>	First character after the last slash before the specified position.
out	<i>min</i>	First digit of the number around the specified position, including preceding zeroes.
out	<i>num</i>	First digit of the number around the specified position, excluding preceding zeroes.

out	max	First position after the number around the specified position.
-----	-----	--

Returns

Value of the number around the specified position.

6.24.3.33 const unsigned int & magrathea::FileList::offset() const [inline]

Offset getter.

Gets the offset between file identificator and list index.

Returns

Immutable reference to the offset.

6.24.3.34 bool magrathea::FileList::operator!= (const FileList & rhs) const [inline]

Not equal to.

Compares for difference and returns true if two elements returned by the subscript operator differs: the internal storage is not compared.

Parameters

in	rhs	Right-hand side.
----	-----	------------------

Returns

True if not equal, false if equal.

6.24.3.35 std::string magrathea::FileList::operator() (const unsigned int i) const [inline]

Generator operator.

Generates the file name using the specified index and ignoring the root, the size limit (except if the list is based on a container) and the offset.

Parameters

in	i	Index.
----	---	--------

Returns

Copy of the generated element.

6.24.3.36 template<class ContainerType , class > FileList & magrathea::FileList::operator= (const ContainerType & source) [inline]

Container assignment operator.

Assigns the file list contents using a container of paths and without any root prefix.

Template Parameters

<i>ContainerType</i>	(Container type.)
----------------------	-------------------

Parameters

in	<i>source</i>	Source of the copy.
----	---------------	---------------------

Returns

Self reference.

6.24.3.37 template<class FormatType , class , class > FileList & magrathea::FileList::operator= (const FormatType & *cformat*) [inline]

Format assignment operator.

Assigns the file list contents using a C-like format string without any root prefix and with a size of one element.

Template Parameters

<i>FormatType</i>	(Type of the format string.)
-------------------	------------------------------

Parameters

in	<i>cformat</i>	Format of file names.
----	----------------	-----------------------

Returns

Self reference.

6.24.3.38 template<class Function , class , class , class > FileList & magrathea::FileList::operator= (Function && *f*) [inline]

Generator assignment operator.

Assigns the file list contents usign a generator function without any root prefix and with a size of one element.

Template Parameters

<i>Function</i>	(Function type : std::string (const unsigned int).)
-----------------	---

Parameters

in	<i>f</i>	Function object std::string (const unsigned int).
----	----------	---

Returns

Self reference.

6.24.3.39 bool magrathea::FileList::operator== (const FileList & *rhs*) const [inline]

Equal to.

Compares for equality and returns true if all elements returned by the subscript operator are equal: the internal storage is not compared and consequently a container-based file list can be equal to a format-based file list.

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

True if equal, false if not equal.

6.24.3.40 std::string magrathea::FileList::operator[](const unsigned int *i*) const [inline]

Subscript operator.

Returns the file name at the specified position in the file list or returns an empty string if it does not exist.

Parameters

in	<i>i</i>	Index.
----	----------	--------

Returns

Copy of the specified element.

6.24.3.41 FileList & magrathea::FileList::reserve(const unsigned int *amount*) [inline]

Reserve space for the container.

Increases the capacity of the container if the specified size is greater than the current one.

Parameters

in	<i>amount</i>	New capacity.
----	---------------	---------------

Returns

Self reference.

6.24.3.42 FileList & magrathea::FileList::resize(const unsigned int *amount*) [inline]

Resize.

Resizes the current list and returns it. In case of a container based list, the new contents is initialized as empty. Note that setting a null size, empties the list but does not clear it.

Parameters

in	<i>amount</i>	New size.
----	---------------	-----------

Returns

Self reference.

6.24.3.43 const std::string & magrathea::FileList::root() const [inline]

Root getter.

Gets the common root prefix.

Returns

Immutable reference to the root.

6.24.3.44 `FileList & magrathea::FileList::shrink() [inline]`

Shrink the container.

Shrinks the container capacity to fit its size.

Returns

Self reference.

6.24.3.45 `const unsigned int & magrathea::FileList::size() const [inline]`

Size getter.

Gets the total file list size.

Returns

Immutable reference to the size.

6.24.3.46 `template<class Function , class > fileList & magrathea::FileList::sort(Function && f=Function())`

Sort file names.

Sorts the file names using the provided comparator function if the file list is based on a container. If the file list is based on a format, it is checked whether the two first files are sorted and an error is thrown if they are not. Finally, if the file list is based on a generator function, the order is checked and an error is thrown in case of non sorted list.

Template Parameters

<i>Function</i>	(Function type : <code>bool(std::string, std::string)</code> .)
-----------------	---

Parameters

<code>in</code>	<code>f</code> Function object <code>bool(std::string, std::string)</code> .
-----------------	--

Returns

Self reference.

Exceptions

<code>std::runtime_error</code>	Generator based list cannot be modified.
<code>std::runtime_error</code>	<code>Constant</code> format based list cannot be modified.

6.24.3.47 `template<class Function , class > bool magrathea::FileList::sorted(Function && f=Function()) const`

Check sorting.

Checks whether the file names are sorted using the provided comparator function.

Template Parameters

<i>Function</i>	(Function type : bool (std::string, std::string) .)
-----------------	---

Parameters

in	<i>f</i> Function object bool (std::string, std::string).
----	---

Returns

True if the contents is sorted, false otherwise.

6.24.3.48 template<class Function , class > bool magrathea::FileList::unicity (Function && f=Function ()) const

Check unicity.

Checks whether the file names are unique using the provided comparator function.

Template Parameters

<i>Function</i>	(Function type : bool (std::string, std::string) .)
-----------------	---

Parameters

in	<i>f</i> Function object bool (std::string, std::string).
----	---

Returns

True if the contents is unique, false otherwise.

6.24.3.49 template<class Function , class > FileList & magrathea::FileList::unique (Function && f=Function ())

Unique file names.

Erases consecutive equal file names using the provided comparator function if the file list is based on a container. If the file list is based on a format, it is checked whether the two first files are unique and if they are not, the file size is set equals to one. Finally, if the file list is based on a generator function, the unicity is checked and an error is thrown in case of non unique names.

Template Parameters

<i>Function</i>	(Function type : bool (std::string, std::string) .)
-----------------	---

Parameters

in	<i>f</i> Function object bool (std::string, std::string).
----	---

Returns

Self reference.

Exceptions

<i>std::runtime_error</i>	Generator based list cannot be modified.
---------------------------	--

6.24.4 Friends And Related Function Documentation

6.24.4.1 std::ostream& operator<< (std::ostream & lhs, const FileList & rhs) [friend]

Output stream operator.

Adds each file name to the stream using the filling character as a separator.

Parameters

in, out	<i>lhs</i>	Left-hand side stream.
in	<i>rhs</i>	Right-hand side file list.

Returns

Output stream.

6.24.5 Member Data Documentation

6.24.5.1 std::vector<std::string> magrathea::FileList::_container

Container of file names.

6.24.5.2 std::string magrathea::FileList::_format

Format of file names.

6.24.5.3 std::function<std::string(const unsigned int)> magrathea::FileList::_generator

Generator of file names.

6.24.5.4 int magrathea::FileList::_length

Optional width of the format field.

6.24.5.5 unsigned int magrathea::FileList::_offset

Offset between file identifier and list index.

6.24.5.6 std::string magrathea::FileList::_root

Common root prefix.

6.24.5.7 unsigned int magrathea::FileList::_size

Total file list size.

The documentation for this exception was generated from the following file:

- /data/home/mbreton/mgrathea_pathfinder/src/mgrathea/filelist.h

6.25 magrathea::FileSystem Exception Reference

Global file management.

```
#include <filesystem.h>
```

Static Public Member Functions

Utilities

- static bool **endianness** ()

Get system endianness.
- template<bool Byteswap, typename Type , class = typename std::enable_if<!Byteswap>::type>
 static bool **byteswap** (const Type &variable)

Do not swap bytes.
- template<bool Byteswap = true, typename Type , class = typename std::enable_if<Byteswap>::type>
 static bool **byteswap** (Type &variable)

Swap bytes.
- template<typename Type = unsigned int, class = typename std::enable_if<std::is_integral<Type>::value>::type>
 static Type **bom** ()

Byte order mark.
- template<typename Type , class = typename std::enable_if<std::is_integral<Type>::value>::type>
 static bool **bom** (const Type &mark)

Byte order mark endianness.

Size

- template<typename Type = char, class Stream >
 static long long int **weight** (Stream &stream)

Get the weight of a file in terms of a specific type.
- static long long int **size** (const std::string &filename)

Get size of a file passed by name.
- static long long int **size** (std::istream &stream)

Get size of a file passed by input stream.
- static long long int **size** (std::ostream &stream)

Get size of a file passed by output stream.

Management

- static std::string **temporary** (const std::string &prefix="\b", const std::string &suffix="")

Temporary file name.
- static std::string **dated** (const std::string &prefix="", const std::string &suffix="", const std::string &format="%Y-%m-%d-%H-%M-%S")

Dated file name.
- static bool **remove** (const std::string &filename)

Remove a file.
- static bool **rename** (const std::string &oldname, const std::string &newname, const std::ios::openmode &mode=std::ios::out)

Rename a file.
- static bool **copy** (const std::string &oldname, const std::string &newname, const std::ios::openmode &mode=std::ios::out, const long long int chunk=-1)

Copy.

Split and join

- template<bool Byteswap = false, typename Marker = long long int, class Container >
 static unsigned int **split** (const std::string &filename, const Container &filenames, const std::ios::openmode &mode=std::ios::out, const long long int chunk=-1, const unsigned long long int limit=0)

Split a file in several files.

- template<bool Byteswap = false, typename Marker = long long int, class Container >
static unsigned int **unsplit** (const Container &filenames, const std::string &filename, const std::ios::openmode &mode=std::ios::out, const long long int chunk=-1)

Unsplit several files in a file.

- template<bool Byteswap = false, typename Marker = long long int, class Container >
static unsigned int **join** (const Container &filenames, const std::string &filename, const std::ios::openmode &mode=std::ios::out, const long long int chunk=-1)

Join several files in a file.

- template<bool Byteswap = false, typename Marker = long long int, class Container >
static unsigned int **unjoin** (const std::string &filename, const Container &filenames, const std::ios::openmode &mode=std::ios::out, const long long int chunk=-1)

Unjoin a file in several files.

Comparison

- static bool **compare** (const std::string &first, const std::string &second, const long long int chunk=-1)

Compare two files passed by names.

- static bool **compare** (std::istream &first, std::istream &second, const long long int chunk=-1)

Compare two files passed by input stream.

- static bool **compare** (std::ostream &first, std::ostream &second, const long long int chunk=-1)

Compare two files passed by output stream.

Existence and creation

- static bool **exist** (const std::string &filename)

Check file existence.

- template<typename Type = char>

static bool **check** (const std::string &filename, const long long int min=-1, const long long int max=-1)

Check file consistency.

- template<class Contents = std::true_type>

static bool **create** (const std::string &filename, const **Contents** &contents=**Contents**())

Create a file.

- template<class Contents = std::true_type>

static bool **initialize** (const std::string &filename, const **Contents** &contents=**Contents**())

Initialize a new file.

- template<class Contents = std::true_type>

static bool **reset** (const std::string &filename, const **Contents** &contents=**Contents**())

Reset an existing file.

- template<class Contents = char, class = typename std::enable_if<!std::is_convertible<Contents, std::string>::value>::type>

static bool **generate** (const std::string &filename, const std::ios::openmode &mode, const long long int amount, const long long int chunk=-1, const **Contents** &contents=**Contents**())

Generate a binary file based on contents.

- template<class Contents , class = typename std::enable_if<std::is_convertible<Contents, std::string>::value>::type, class = void>

static bool **generate** (const std::string &filename, const std::ios::openmode &mode, const long long int amount, const long long int chunk, const **Contents** &contents)

- template<class Engine , class Distribution , class Contents = typename std::decay<Distribution>::type::result_type, class = typename std::enable_if<(std::decay<Engine>::type::min() != std::decay<Engine>::type::max()) && (!std::is_void<typename std::decay<Distribution>::type::result_type>::value)>::type>

static bool **generate** (const std::string &filename, const std::ios::openmode &mode, const long long int amount, const long long int chunk, Engine &&engine, Distribution &&distribution)

Generate a random binary file.

Size control

- template<typename Type = char, class Stream >

static bool **empty** (Stream &stream)

Empty file.

- template<typename Type = char, class Stream >
static bool **exact** (Stream &stream, const long long int amount)
File of exact specified size.
- template<typename Type = char, class Stream >
static bool **regular** (Stream &stream, const long long int min=-1, const long long int max=-1)
Regular file.

File types

- static bool **ascii** (const std::string &filename, const long long int min=-1, const long long int max=-1, const long long int chunk=-1)
Ascii file.
- static bool **eascii** (const std::string &filename, const long long int min=-1, const long long int max=-1, const long long int chunk=-1)
Extended ascii file.
- template<typename Type = char>
static bool **binary** (const std::string &filename, const long long int min=-1, const long long int max=-1, const long long int chunk=-1)
Binary file.

Test

- static int **example** ()
Example function.

6.25.1 Detailed Description

Global file management.

Provides general functions to manage files, retrieve some information and perform standard operations on files. All functions are standard-compliant, but use either C++ or C depending on performances. Most functions return whether the operation is a success and do not throw any exception. In the class, an unit means the size in bytes of a provided type.

6.25.2 Member Function Documentation

6.25.2.1 bool magrathea::FileSystem::ascii (const std::string & filename, const long long int min = -1, const long long int max = -1, const long long int chunk = -1) [static]

Ascii file.

Tests whether the file is an ascii file containing between the minimum and maximum amount of bytes and using the chunk parameter to control the buffer size. A file is considered as ascii if all its bytes are in the ranges [9, 13] or [32, 126]. An empty file is considered as an ascii file. A value of -1 for the minimum or for the maximum means that this boundary is not tested. Finally, the chunk parameter allows to specify whether a buffer is used in binary mode : 0 corresponds to no buffer, a positive integer corresponds to the amount of contents that is put into the buffer, and a negative integer corresponds to a buffer of the total size of the file.

Parameters

in	<i>filename</i>	File name.
in	<i>min</i>	Check if the file contains at least this amount of data.
in	<i>max</i>	Check if the file contains at most this amount of data.
in	<i>chunk</i>	Buffer size.

Returns

True if the file is an ascii file satisfying the provided conditions, false otherwise.

6.25.2.2 template<typename Type > bool magrathea::FileSystem::binary (const std::string & *filename*, const long long int *min* = -1, const long long int *max* = -1, const long long int *chunk* = -1) [static]

Binary file.

Tests whether the file is a binary file containing between the minimum and maximum amount of data and using the chunk parameter to control the buffer size. A file is considered as binary if it has a byte outside of the ranges [9, 13] or [32, 126], or [128, 255]. An empty file is not considered as a binary file. If the file size is not divisible by the specified unit, the function returns false. A value of -1 for the minimum or for the maximum means that this boundary is not tested. Finally, the chunk parameter allows to specify whether a buffer is used in binary mode : 0 corresponds to no buffer, a positive integer corresponds to the amount of contents that is put into the buffer, and a negative integer corresponds to a buffer of the total size of the file.

Parameters

in	<i>filename</i>	File name.
in	<i>min</i>	Check if the file contains at least this amount of data.
in	<i>max</i>	Check if the file contains at most this amount of data.
in	<i>chunk</i>	Buffer size.

Returns

True if the file is a binary file satisfying the provided conditions, false otherwise.

6.25.2.3 template<typename Type , class > Type magrathea::FileSystem::bom () [inline], [static]

Byte order mark.

Returns the byte order mark 0xFEFF casted to the provided type. It is used to detect the endianness : for example, for a 4-bytes integer : 00-00-FE-FF indicates big-endianness and FF-FE-00-00 indicates little-endianness.

Template Parameters

Type	Integral byte order mark type.
------	--------------------------------

Returns

Cast of 0xFEFF.

6.25.2.4 template<typename Type , class > bool magrathea::FileSystem::bom (const Type & *mark*) [static]

Byte order mark endianness.

Returns endianness from byte order mark passed as parameter : true for big-endian, false for little-endian. The byte order mark should be of the form FE-FF.

Template Parameters

Type	(Integral byte order mark type.)
------	----------------------------------

Parameters

<i>in</i>	<i>mark</i>	Byte order mark.
-----------	-------------	------------------

Returns

True for big-endian, false for little-endian.

Exceptions

<i>std::invalid_argument</i>	Unrecognized byte order mark.
------------------------------	-------------------------------

6.25.2.5 template<bool Byteswap, typename Type , class > bool magrathea::FileSystem::byteswap (const Type & variable) [inline], [static]

Do not swap bytes.

Does not invert the order of bytes of the passed variable to keep the endianness.

Template Parameters

<i>Byteswap</i>	Do not swap endianness if false.
<i>Type</i>	(Variable type.)

Parameters

<i>in</i>	<i>variable</i>	Variable.
-----------	-----------------	-----------

Returns

False if the variable has not been byteswapped.

6.25.2.6 template<bool Byteswap, typename Type , class > bool magrathea::FileSystem::byteswap (Type & variable) [inline], [static]

Swap bytes.

Inverts the order of bytes of the passed variable to change the endianness.

Template Parameters

<i>Byteswap</i>	Swap endianness if true.
<i>Type</i>	(Variable type.)

Parameters

<i>in, out</i>	<i>variable</i>	Variable.
----------------	-----------------	-----------

Returns

True if the variable has been byteswapped.

6.25.2.7 template<typename Type > bool magrathea::FileSystem::check (const std::string & *filename*, const long long int *min* = -1, const long long int *max* = -1) [static]

Check file consistency.

Returns whether the file can be opened successfully and whether it contains between the minimum and the maximum data of the provided type. If the total size is not divisible by the size of the type, the function returns false. If the file is not empty, reading a byte is also tested. A value of -1 for the minimum or for the maximum means that this boundary is not tested.

Template Parameters

Type	Type representing the considered unit.
------	--

Parameters

in	<i>filename</i>	File name.
in	<i>min</i>	Check if the file contains at least this amount of data.
in	<i>max</i>	Check if the file contains at most this amount of data.

Returns

True if the file can be opened and is compliant to the provided parameters, false otherwise.

6.25.2.8 bool magrathea::FileSystem::compare (const std::string & *first*, const std::string & *second*, const long long int *chunk* = -1) [static]

Compare two files passed by names.

Compares two files from their file names and return true if both can be tested without errors, have the same size, and have the same contents. A file compared to itself returns true whether it can be opened without errors. Finally, the chunk parameter allows to specify whether a buffer is used for the comparison : 0 corresponds to no buffer, a positive integer corresponds to the amount of bytes that is put into the buffer, and a negative integer corresponds to a buffer of the total size of the file.

Parameters

in	<i>first</i>	First file name.
in	<i>second</i>	Second file name.
in	<i>chunk</i>	Buffer size.

Returns

True if both files compares equal without errors, false otherwise.

6.25.2.9 bool magrathea::FileSystem::compare (std::istream & *first*, std::istream & *second*, const long long int *chunk* = -1) [static]

Compare two files passed by input stream.

Compares two opened input streams and return true if both can be tested without errors, have the same size, and have the same contents. A stream compared to itself returns true whether it can be accessed without errors. Finally,

the chunk parameter allows to specify whether a buffer is used for the comparison : 0 corresponds to no buffer, a positive integer corresponds to the amount of bytes that is put into the buffer, and a negative integer corresponds to a buffer of the total size of the file.

Parameters

<code>in,out</code>	<code>first</code>	First input stream.
<code>in,out</code>	<code>second</code>	Second input stream.
<code>in</code>	<code>chunk</code>	Buffer size.

Returns

True if both input streams compares equal without errors, false otherwise.

6.25.2.10 `bool magrathea::FileSystem::compare (std::ostream & first, std::ostream & second, const long long int chunk = -1) [static]`

Compare two files passed by output stream.

Compares two opened output streams and return true if both can be tested without errors and have the same size as no further comparison can be done with output streams. Finally, the chunk parameter allows to specify whether a buffer is used for the comparison : 0 corresponds to no buffer, a positive integer corresponds to the amount of bytes that is put into the buffer, and negative integer corresponds to a buffer of the total size of the file.

Parameters

<code>in,out</code>	<code>first</code>	First output stream.
<code>in,out</code>	<code>second</code>	Second output stream.
<code>in</code>	<code>chunk</code>	Buffer size.

Returns

True if both output streams compares equal without errors, false otherwise.

6.25.2.11 `bool magrathea::FileSystem::copy (const std::string & oldname, const std::string & newname, const std::ios::openmode & mode = std::ios::out, const long long int chunk = -1) [static]`

Copy.

Copy the provided file to a new location. If the old and new names are equal, the function returns false without further tests. By default, if a standard output open mode is specified, all existing files are protected and only a new one can be created. This protection can be removed by explicitly specifying the truncate open mode. Finally, the chunk parameter allows to specify whether a buffer is used for the copy : 0 corresponds to no buffer, a positive integer corresponds to the amount of bytes that is put into the buffer, and a negative integer corresponds to a buffer of the total size of the file.

Parameters

<code>in</code>	<code>oldname</code>	Old file name.
<code>in</code>	<code>newname</code>	New file name.
<code>in</code>	<code>mode</code>	Open mode.
<code>in</code>	<code>chunk</code>	Buffer size.

Returns

True if the copy was done without errors, false otherwise.

6.25.2.12 template<class **Contents**> bool magrathea::FileSystem::create (const std::string & *filename*, const **Contents** & *contents* = **Contents** ()) [static]

Create a file.

Creates a file, overwriting any previous file if needed. This is equivalent to create a new file using an `std::ofstream` and the `std::ios::trunc` open mode. The extra parameter allows to initialize the file with contents : if this parameter is convertible to a string, the file is opened as a text file, otherwise it is considered as a binary file.

Template Parameters

<i>Contents</i>	<i>Contents</i> type.
-----------------	-----------------------

Parameters

in	<i>filename</i>	File name.
in	<i>contents</i>	Contents to add to the file.

Returns

True if the file was created without errors, false otherwise.

6.25.2.13 std::string magrathea::FileSystem::dated (const std::string & *prefix* = " ", const std::string & *suffix* = " ", const std::string & *format* = "%Y-%m-%d-%H-%M-%S") [inline], [static]

Dated file name.

Generates a file name from the current time.

Parameters

in	<i>prefix</i>	File name prefix or path.
in	<i>suffix</i>	File name suffix or extension.
in	<i>format</i>	Date format compatible with <code>strftime()</code> .

Returns

Generated dated file name.

6.25.2.14 bool magrathea::FileSystem::eascii (const std::string & *filename*, const long long int *min* = -1, const long long int *max* = -1, const long long int *chunk* = -1) [static]

Extended ascii file.

Tests whether the file is an extended ascii file containing between the minimum and maximum amount of bytes and using the chunk parameter to control the buffer size. A file is considered as extended ascii if all its bytes are in the ranges [9, 13] or [32, 126], or [128, 255]. An empty file is considered as an extended ascii file. A value of -1 for the minimum or for the maximum means that this boundary is not tested. Finally, the chunk parameter allows to specify whether a buffer is used in binary mode : 0 corresponds to no buffer, a positive integer corresponds to the amount of contents that is put into the buffer, and a negative integer corresponds to a buffer of the total size of the file.

Parameters

in	<i>filename</i>	File name.
in	<i>min</i>	Check if the file contains at least this amount of data.
in	<i>max</i>	Check if the file contains at most this amount of data.
in	<i>chunk</i>	Buffer size.

Returns

True if the file is an extended ascii file satisfying the provided conditions, false otherwise.

6.25.2.15 template<typename Type , class Stream > bool magrathea::FileSystem::empty (Stream & stream) [inline], [static]

Empty file.

Returns whether the file exists and is empty. The unit type is here for compatibility reasons with other size control functions.

Template Parameters

Type	Type representing the considered unit.
Stream	(String, input stream or output stream.)

Parameters

in, out	stream	Stream.
---------	--------	---------

Returns

True if the file size is null, false otherwise.

6.25.2.16 bool magrathea::FileSystem::endianness () [inline], [static]

Get system endianness.

Returns the system endianness tested with an integer.

Returns

True for big-endian, false for little-endian.

6.25.2.17 template<typename Type , class Stream > bool magrathea::FileSystem::exact (Stream & stream, const long long int amount) [inline], [static]

File of exact specified size.

Returns whether the file size measured in the specified unit is exactly equals to the value. For example if the specified type is an integer, and the value is equal to 4, it returns true if the file contains exactly 4 integers. A value equals to -1 returns true if the file size cannot be computed.

Template Parameters

Type	Type representing the considered unit.
Stream	(String, input stream or output stream.)

Parameters

in, out	stream	Stream.
in	amount	Amount of contents to for comparison.

Returns

True if the file size corresponds to the provided value, false otherwise.

6.25.2.18 int magrathea::FileSystem::example() [static]

Example function.

Tests and demonstrates the use of [FileSystem](#).

Returns

0 if no error.

6.25.2.19 bool magrathea::FileSystem::exist(const std::string & *filename*) [inline], [static]

Check file existence.

Returns whether the file can be opened. The difference with the [check\(\)](#) function is that [exists\(\)](#) use a C test without checking the C++ error bits. Consequently, it may be faster than the [check\(\)](#) implementation and therefore is well suited to check the existence of a large number of files.

Parameters

<code>in</code>	<code>filename</code>	File name.
-----------------	-----------------------	------------

Returns

True if the file exists, false otherwise.

6.25.2.20 template<class Contents , class , class > bool magrathea::FileSystem::generate(const std::string & *filename*, const std::ios::openmode & *mode*, const long long int *amount*, const long long int *chunk* = -1, const Contents & *contents* = *Contents* ()) [static]

Generate a binary file based on contents.

Generate a text file based on contents.

Creates a new file putting a repetition of the specified amount of contents in it. To use the binary mode, this contents should not be convertible to a string. By default, if a standard output open mode is specified, all existing files are protected and only a new one can be created. This protection can be removed by explicitly specifying the truncate open mode. Finally, the chunk parameter allows to specify whether a buffer is used in binary mode : 0 corresponds to no buffer, a positive integer corresponds to the amount of contents that is put into the buffer, and a negative integer corresponds to a buffer of the total size of the file.

Template Parameters

Contents	(Contents type.)
--------------------------	------------------

Parameters

<code>in</code>	<code>filename</code>	File name.
<code>in</code>	<code>mode</code>	Open mode.
<code>in</code>	<code>amount</code>	Amount of contents to be put in the file.
<code>in</code>	<code>chunk</code>	Buffer size.
<code>in</code>	<code>contents</code>	Contents to add to the file.

Returns

True if the file was created without errors, false otherwise.

Creates a new text file putting a repetition of the specified amount of contents in it. To use the text mode, this contents should be convertible to a string. By default, if a standard output open mode is specified, all existing files are protected and only a new one can be created. This protection can be removed by explicitly specifying the truncate open mode. Finally, the chunk parameter allows to specify whether a buffer string is used in text mode : 0 corresponds to no buffer, a positive integer corresponds to the amount of contents copies that are put into the buffer, and a negative integer corresponds to a buffer of the total size of the file.

Template Parameters

Contents	(Contents type.)
--------------------------	------------------

Parameters

in	<i>filename</i>	File name.
in	<i>mode</i>	Open mode.
in	<i>amount</i>	Amount of contents to be put in the file.
in	<i>chunk</i>	Buffer size.
in	<i>contents</i>	Contents to add to the file.

Returns

True if the file was created without errors, false otherwise.

6.25.2.21 `template<class Contents , class = typename std::enable_if<std::is_convertible<Contents, std::string>::value>::type, class = void> static bool magrathea::FileSystem::generate (const std::string & filename, const std::ios::openmode & mode, const long long int amount, const long long int chunk, const Contents & contents) [static]`

6.25.2.22 `template<class Engine , class Distribution , class Contents , class > bool magrathea::FileSystem::generate (const std::string & filename, const std::ios::openmode & mode, const long long int amount, const long long int chunk, Engine && engine, Distribution && distribution) [static]`

Generate a random binary file.

Creates a new random file putting a repetition of the random numbers generated thanks to the specified engine and distribution and using the distribution result type as the contents type. By default, if a standard output open mode is specified, all existing files are protected and only a new one can be created. This protection can be removed by explicitly specifying the truncate open mode. Finally, the chunk parameter allows to specify whether a buffer is used in binary mode : 0 corresponds to no buffer, a positive integer corresponds to the amount of contents that is put into the buffer, and a negative integer corresponds to a buffer of the total size of the file.

Template Parameters

<i>Engine</i>	(Random engine type.)
<i>Distribution</i>	(Random distribution type.)
<i>Contents</i>	(Contents type.)

Parameters

in	<i>filename</i>	File name.
in	<i>mode</i>	Open mode.
in	<i>amount</i>	Amount of contents to be put in the file.
in	<i>chunk</i>	Buffer size.
in,out	<i>engine</i>	Random engine.

in, out	<i>distribution</i>	Random distribution.
---------	---------------------	----------------------

Returns

True if the file was created without errors, false otherwise.

6.25.2.23 template<class **Contents> bool magrathea::FileSystem::initialize (const std::string & *filename*, const **Contents** & *contents* = **Contents** ()) [static]**

Initialize a new file.

Creates a new file, without overwriting any previous file. This function is well suited to create a file without risking to erase some important existing data. The extra parameter allows to initialize the file with contents : if this parameter is convertible to a string, the file is opened as a text file, otherwise it is considered as a binary file.

Template Parameters

<i>Contents</i>	<i>Contents</i> type.
-----------------	-----------------------

Parameters

in	<i>filename</i>	File name.
in	<i>contents</i>	<i>Contents</i> to add to the file.

Returns

True if the file was created without errors, false otherwise.

6.25.2.24 template<bool **Byteswap, typename **Marker**, class **Container**> unsigned int magrathea::FileSystem::join (const **Container** & *filenames*, const std::string & *filename*, const std::ios::openmode & *mode* = std::ios::out, const long long int *chunk* = -1) [static]**

Join several files in a file.

Joins the provided list of file into a single file. At the beginning and the end of each file a record marker of the file size in bytes is added. By default, if a standard output open mode is specified, all existing files are protected and only a new one can be created. This protection can be removed by explicitly specifying the truncate open mode. Finally, the chunk parameter allows to specify whether a buffer is used for the copy : 0 corresponds to no buffer, a positive integer corresponds to the amount of bytes that is put into the buffer, and a negative integer corresponds to a buffer of the total size of the file.

Template Parameters

<i>Byteswap</i>	Swap endianness of markers.
<i>Marker</i>	Record marker type.
<i>Container</i>	(File container type.)

Parameters

in	<i>filenames</i>	<i>Input</i> file names to join.
in	<i>filename</i>	<i>Output</i> file name.
in	<i>mode</i>	Open mode.
in	<i>chunk</i>	Buffer size.

Returns

Number of files written on success, zero on error.

6.25.2.25 template<typename Type , class Stream > bool magrathea::FileSystem::regular (Stream & stream, const long long int min = -1, const long long int max = -1) [inline], [static]

Regular file.

Tests if the file is regular regarding to the provided options. It returns true if the file can be read without problems, if its size in bytes can be divided by the size of the provided type, and if its size in the specified unit is between the minimum and maximum provided amount of data. A value of -1 for the minimum or for the maximum means that this boundary is not tested.

Template Parameters

Type	Type representing the considered unit.
Stream	(String, input stream or output stream.)

Parameters

in, out	stream	Stream.
in	min	Check if the file contains at least this amount of data.
in	max	Check if the file contains at most this amount of data.

Returns

True if the file size is regular regarding the parameters, false otherwise.

6.25.2.26 bool magrathea::FileSystem::remove (const std::string & filename) [inline], [static]

Remove a file.

Removes an existing file from the file system and returns true on success.

Parameters

in	filename	File name.
----	----------	------------

Returns

True on success, false on error.

6.25.2.27 bool magrathea::FileSystem::rename (const std::string & oldname, const std::string & newname, const std::ios::openmode & mode = std::ios::out) [inline], [static]

Rename a file.

Renames an existing file and returns true on success. If the old and new names are equal, the function returns false without further tests. If the new file name already exists, nothing is done and the function fails except if the truncate open mode is specified : in that case, the existing file is erased.

Parameters

in	oldname	Old name.
in	newname	New name.
in	mode	Open mode.

Returns

True on success, false on error.

6.25.2.28 template<class Contents > bool magrathea::FileSystem::reset (const std::string & *filename*, const Contents & *contents* = Contents ()) [static]

Reset an existing file.

Erases the contents of an existing file without creating a new one if the specified name does not exist. This function is well suited to avoid the unexpected creation of new files. The extra parameter allows to initialize the file with contents : if this parameter is convertible to a string, the file is opened as a text file, otherwise it is considered as a binary file.

Template Parameters

<i>Contents</i>	Contents type.
-----------------	----------------

Parameters

in	<i>filename</i>	File name.
in	<i>contents</i>	Contents to add to the file.

Returns

True if the file was created without errors, false otherwise.

6.25.2.29 long long int magrathea::FileSystem::size (const std::string & *filename*) [static]

Get size of a file passed by name.

Opens the file, computes its size and closes it. If the file does not exist or if a stream error is detected, -1 is returned.

Parameters

in	<i>filename</i>	File name.
----	-----------------	------------

Returns

File size in bytes or -1 if error.

6.25.2.30 long long int magrathea::FileSystem::size (std::istream & *stream*) [static]

Get size of a file passed by input stream.

Saves the current position, computes the size of the passed stream and returns to the original position. If the file does not exist or if a stream error is detected, -1 is returned.

Parameters

in, out	<i>stream</i>	Input stream.
---------	---------------	---------------

Returns

File size in bytes or -1 if error.

6.25.2.31 long long int magrathea::FileSystem::size (std::ostream & stream) [static]

Get size of a file passed by output stream.

Saves the current position, computes the size of the passed stream and returns to the original position. If the file does not exist or if a stream error is detected, -1 is returned.

Parameters

in, out	<i>stream</i>	Output stream.
---------	---------------	----------------

Returns

File size in bytes or -1 if error.

6.25.2.32 template<bool Byteswap, typename Marker , class Container > unsigned int magrathea::FileSystem::split (const std::string & filename, const Container & filenames, const std::ios::openmode & mode = std::ios::out, const long long int chunk = -1, const unsigned long long int limit = 0) [static]

Split a file in several files.

Splits the provided file into several ones of lower sizes. At the beginning and the end of each file, the current iterator byte position regarding the splitted file is saved as a marker. A beginning marker equals to zero corresponds to the first and an end marker equals to zero corresponds to the last file. The limit parameter allows to limit the size by file to an exact number of bytes. By default, if a standard output open mode is specified, all existing files are protected and only new ones can be created. This protection can be removed by explicitly specifying the truncate open mode. Finally, the chunk parameter allows to specify whether a buffer is used for the copy : 0 corresponds to no buffer, a positive integer corresponds to the amount of bytes that is put into the buffer, and a negative integer corresponds to a buffer of the total size of the file.

Template Parameters

<i>Byteswap</i>	Swap endianness of markers.
<i>Marker</i>	Record marker type.
<i>Container</i>	(File container type.)

Parameters

in	<i>filename</i>	Input file name to split.
in	<i>filenames</i>	Output file names.
in	<i>mode</i>	Open mode.
in	<i>chunk</i>	Buffer size.
in	<i>limit</i>	Size limit of each file.

Returns

Number of files written on success, zero on error.

6.25.2.33 std::string magrathea::FileSystem::temporary (const std::string & prefix = "\b", const std::string & suffix = " ") [inline], [static]

Temporary file name.

Generates a temporary file name. If no argument is used, then the default location is used. If an empty prefix is specified, the path is erased, and only the file name is kept. If a prefix or a suffix is specified, the default path is erased, the file name is kept and prefixed and suffixed by the arguments.

Parameters

in	<i>prefix</i>	File name prefix or path.
in	<i>suffix</i>	File name suffix or extension.

Returns

Generated temporary file name.

```
6.25.2.34 template<bool Byteswap, typename Marker , class Container > unsigned int magrathea::FileSystem::unjoin ( const std::string & filename, const Container & filenames, const std::ios::openmode & mode = std::ios::out, const long long int chunk = -1 ) [static]
```

Unjoin a file in several files.

Unjoins the provided file into the original ones. The original marker size should correspond to the provided one. If output file names are not unique, the previous file is truncated or not depending on the specified open mode. By default, if a standard output open mode is specified, all existing files are protected and only new ones can be created. This protection can be removed by explicitly specifying the truncate open mode. Finally, the chunk parameter allows to specify whether a buffer is used for the copy : 0 corresponds to no buffer, a positive integer corresponds to the amount of bytes that is put into the buffer, and a negative integer corresponds to a buffer of the total size of the file.

Template Parameters

<i>Byteswap</i>	Swap endianness of markers.
<i>Marker</i>	Record marker type.
<i>Container</i>	(File container type.)

Parameters

in	<i>filename</i>	Input file name to unjoin.
in	<i>filenames</i>	Output file names.
in	<i>mode</i>	Open mode.
in	<i>chunk</i>	Buffer size.

Returns

Number of files written on success, zero on error.

```
6.25.2.35 template<bool Byteswap, typename Marker , class Container > unsigned int magrathea::FileSystem::unspli ( const Container & filenames, const std::string & filename, const std::ios::openmode & mode = std::ios::out, const long long int chunk = -1 ) [static]
```

Unsplit several files in a file.

Unsplits the provided list of files into the original one. The original marker size should correspond to the provided one. By default, if a standard output open mode is specified, all existing files are protected and only a new one can be created. This protection can be removed by explicitly specifying the truncate open mode. Finally, the chunk parameter allows to specify whether a buffer is used for the copy : 0 corresponds to no buffer, a positive integer corresponds to the amount of bytes that is put into the buffer, and a negative integer corresponds to a buffer of the total size of the file.

Template Parameters

<i>Byteswap</i>	Swap endianness of markers.
<i>Marker</i>	Record marker type.
<i>Container</i>	(File container type.)

Parameters

in	<i>filenames</i>	Input file names to unsplit.
in	<i>filename</i>	Output file name.
in	<i>mode</i>	Open mode.
in	<i>chunk</i>	Buffer size.

Returns

Number of files written on success, zero on error.

6.25.2.36 template<typename Type , class Stream > long long int magrathea::FileSystem::weight (Stream & stream) [inline], [static]

Get the weight of a file in terms of a specific type.

Returns the amount of data of the specified type equivalent to the file size. This is just the file size divided by the size of the specified type rounded to the lower integer. If the file does not exist or if a stream error is detected, -1 is returned.

Template Parameters

Type	Type representing the considered unit.
Stream	(String, input stream or output stream.)

Parameters

in,out	<i>stream</i>	Stream.
--------	---------------	---------

Returns

File weight in unit or -1 if error.

The documentation for this exception was generated from the following file:

- /data/home/mbreton/magrathea_pathfinder/src/magrathea/filesystem.h

6.26 Gravity< Type, Dimension > Exception Template Reference

[Gravity](#) cell implementation for raytracing.

```
#include <gravity.h>
```

Inheritance diagram for Gravity< Type, Dimension >:



Public Member Functions

Lifecycle

- template<class... Misc> [Gravity](#) (Misc &&...misc)
Explicit generic constructor.

- template<class... Misc>
Gravity (Misc &&...misc)

Data

- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<magrathea::AbstractContents<Gravity<-Type, Dimension>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type>>().template data<0, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type>
Template rho (Misc &&...misc)
Access to the rho data.
- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<const magrathea::AbstractContents<-Gravity<Type, Dimension>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type>>().template data<0, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type>
Template rho (Misc &&...misc) const
Immutable access to the rho data.
- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<magrathea::AbstractContents<Gravity<-Type, Dimension>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type>>().template data<1, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type>
Template phi (Misc &&...misc)
Access to the phi data.
- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<const magrathea::AbstractContents<-Gravity<Type, Dimension>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type>>().template data<1, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type>
Template phi (Misc &&...misc) const
Immutable access to the phi data.
- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<magrathea::AbstractContents<Gravity<-Type, Dimension>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type>>().template data<2, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type>
Template dphidxyz (Misc &&...misc)
Access to the dphidxyz data.
- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<const magrathea::AbstractContents<-Gravity<Type, Dimension>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type>>().template data<2, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type>
Template dphidxyz (Misc &&...misc) const
Immutable access to the dphidxyz data.
- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<magrathea::AbstractContents<Gravity<-Type, Dimension>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type>>().template data<2, Values...>(0, std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type>
Template dphidx (Misc &&...misc)
Access to the dphidx data.
- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<const magrathea::AbstractContents<-Gravity<Type, Dimension>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type>>().template data<2, Values...>(0, std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type>
Template dphidx (Misc &&...misc) const
Immutable access to the dphidx data.
- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<magrathea::AbstractContents<Gravity<-Type, Dimension>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type>>().template data<2, Values...>(1, std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type>
Template dphidy (Misc &&...misc)
Access to the dphidy data.
- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<const magrathea::AbstractContents<-Gravity<Type, Dimension>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type>>().template data<2, Values...>(1, std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type>
Template dphidy (Misc &&...misc) const
Immutable access to the dphidy data.
- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<magrathea::AbstractContents<Gravity<-Type, Dimension>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type>>().template data<2, Values...>(2, std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type>
Template dphidz (Misc &&...misc)

Access to the dphidz data.

- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<const magrathea::AbstractContents<-Gravity<Type, Dimension>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type>()).template data<2, Values...>(2, std::declval<Misc>(...)), class = typename std::enable_if<std::is_void<Template>::value>::type> Template **dphidz** (Misc &...misc) const

Immutable access to the dphidz data.

- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<magrathea::AbstractContents<Gravity<-Type, Dimension>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type >()).template data<3, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type>
Template a (Misc &&...misc)

Access to the a data.

plate<unsigned int... > \

- Gravity<Type, Dimension>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type >>().template
data<3, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type>
Template a ([Misc](#) &&...[misc](#)) const

Immutable access to the a data.

plate<unsigned int... Values, class..

- Type, Dimension>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type >>().template data<4, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type>
Template **dphidt** (Misc &...misc)

Access to the dphidt data.

plate<unsigned int... Values

- Gravity<Type, Dimension>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type >>().template
data<4, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<std::is_void<Template>::value>::type>
Template [dphidt](#) (Misc &...misc) const

Immutable access to the aphid

Dimension: months. Editable.

- ```
type, Dimension>, ImaginaryEulerianCategory, type, type, std::array<type, Dimension>, type, type, std::array<type, Dimension>> >().template data<0, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<std::is_void<-Template>::value>::type>
```

update<unsigned int Values class

avity<Type, Dimension>, magrat

- Type, Dimension>> >().template data<0, Values...>(std::declval<Misc>(...)), class = typename std::enable\_if<!std::is\_void<-Template>::value>::type>  
Template [rho](#) (Misc &...misc) const

plate<unsigned int... Values, class... M

e, Dimension>, magrathea::Eule

- Dimension>> >().template data<1, Values...>(std::declval<Misc>()), class = typename std::enable\_if<!std::is\_void<-Template>::value>::type>  
**Template phi (Misc &&...misc)**

plate<unsigned int... Values, cl

Plants are often used to indicate N

- Type, Dimension>> >().template data<1, Values...>(std::decay<Misc>(...)), class = typename std::enable\_if<std::is\_void<-Template>::value>::type>  
Template **phi** (Misc &...misc) const

e Dimension> magrathea::Full

dimension>>>() template data<<

- Template>::value>::type>  
Template [dphidxyz](#) (Misc &&...misc)  
• template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<const magrathea::AbstractContents<-

avity<Type, Dimension>, magrathea::EulerianC

e, Dimension>> >().template data<2, Values.

- Template >::value>::type>  
Template [dphidxyz](#) (Misc &&...misc) const

```
Dimension>> >().template data<2, Values...>(0, std::declval<Misc>(...)), class_ = typename std::enable_if<!std::is_void<-Template>::value>::type>
```

## Template dphidx (Misc &&...misc)

- ```
• template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<const magrathea::AbstractContents<<Gravity<Type, Dimension>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type, std::array<Type, Dimension>>>().template data<2, Values...>(0, std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>>::value>::type>
```

Template `dphidx` (Misc &&...misc) const

- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<magrathea::AbstractContents<Gravity<-Type, Dimension>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type, std::array<Type, Dimension>>> >().template data<2, Values...>(1, std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>>::value>::type>

Template dphidy (Misc &&...misc)

- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<const magrathea::AbstractContents<<Gravity<Type, Dimension>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type, std::array<Type, Dimension>>>().template data<2, Values...>(1, std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>>::value>::type>

Template [dphidy](#) (Misc &&...misc) const

- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<magrathea::AbstractContents<Gravity<<Type, Dimension>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type, std::array<Type, Dimension>>> >().template data<2, Values...>(2, std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>>::value>::type>

Template dphidz (Misc &...misc)

- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<const magrathea::AbstractContents<- Gravity<Type, Dimension>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type, std::array<Type, Dimension>>).template data<2, Values...>(2, std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>>::value>::type>

Template [dphidz](#) (Misc &&...misc) const

- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<magrathea::AbstractContents<Gravity<<Type, Dimension>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type, std::array<Type, Dimension>>>(>).template data<3, Values...>(std::declval<Misc>()...)), class = typename std::enable_if<!std::is_void<Template>>::value>::type>

Template a (Misc &&...misc)

- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<const magrathea::AbstractContents<- Gravity<Type, Dimension>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type, std::array<Type, Dimension>>>().template data<3, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>>::value>::type>

Template a (Misc &...misc) const

- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<magrathea::AbstractContents<Gravity<<Type, Dimension>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type, std::array<Type, Dimension>>> ()>.template data<4, Values...>(std::declval<Misc>()), class = typename std::enable_if<!std::is_void<Template>>::value>::type>

Template dphidt (Misc &&...misc)

- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<const magrathea::AbstractContents<<Gravity<Type, Dimension>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type, std::array<Type, Dimension>>>().template data<4, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>>::value>::type>

Template `dphidt` (Misc &&...misc) const

- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<magrathea::AbstractContents<Gravity<-Type, Dimension>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type, std::array<Type, Dimension>> >().template data<5, Values...>(std::declval<Misc>()), class = typename std::enable_if<!std::is_void<Template>>::value>::type>

Template vxyz (Misc &...misc)

Access to the vxyz data.

- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<const magrathea::AbstractContents<- Gravity<Type, Dimension>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type, std::array<Type, Dimension>>>().template data<5, Values...>(std::declval<Misc>()), class = typename std::enable_if<!std::is_void<Template>>::value>::type>

Template `vxyz` (Misc &&...misc) const

Immutable access to the vxyz data.

- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<magrathea::AbstractContents<Gravity<-Type, Dimension>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type, std::array<Type, Dimension>>>().template data<5, Values...>(0, std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<-Template>::value>::type>
- Template `vx` (Misc &&...misc)**

Access to the vx data.

- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<const magrathea::AbstractContents<-Gravity<Type, Dimension>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type, std::array<Type, Dimension>>>().template data<5, Values...>(0, std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<-Template>::value>::type>

Template `vx` (Misc &&...misc) const

Immutable access to the vx data.

- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<magrathea::AbstractContents<Gravity<-Type, Dimension>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type, std::array<Type, Dimension>>>().template data<5, Values...>(1, std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<-Template>::value>::type>

Template `vy` (Misc &&...misc)

Access to the vy data.

- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<const magrathea::AbstractContents<-Gravity<Type, Dimension>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type, std::array<Type, Dimension>>>().template data<5, Values...>(1, std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<-Template>::value>::type>

Template `vy` (Misc &&...misc) const

Immutable access to the vy data.

- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<magrathea::AbstractContents<Gravity<-Type, Dimension>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type, std::array<Type, Dimension>>>().template data<5, Values...>(2, std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<-Template>::value>::type>

Template `vz` (Misc &&...misc)

Access to the vz data.

- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<const magrathea::AbstractContents<-Gravity<Type, Dimension>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type, std::array<Type, Dimension>>>().template data<5, Values...>(2, std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<-Template>::value>::type>

Template `vz` (Misc &&...misc) const

Immutable access to the vz data.

Static Public Member Functions

Test

- static int `example ()`
Example function.
- static int `example ()`

Public Attributes

- using `operator =` = `typedef`

Additional Inherited Members

6.26.1 Detailed Description

template<typename Type = double, unsigned int Dimension = 3>exception Gravity< Type, Dimension >

`Gravity` cell implementation for raytracing.

A gravity cell containing the local density and the local potential.

Template Parameters

<i>Type</i>	Data type.
<i>Dimension</i>	Number of space dimension.

6.26.2 Constructor & Destructor Documentation

6.26.2.1 template<typename Type , unsigned int Dimension> template<class... Misc> Gravity< Type, Dimension >::Gravity (Misc &&... misc) [inline], [explicit]

Explicit generic constructor.

Provides a generic interface to all constructors of the base class.

Template Parameters

<i>Misc</i>	(Miscellaneous types.)
-------------	------------------------

Parameters

<i>in</i>	<i>misc</i>	Miscellaneous arguments.
-----------	-------------	--------------------------

6.26.2.2 template<typename Type = double, unsigned int Dimension = 3> template<class... Misc> Gravity< Type, Dimension >::Gravity (Misc &&... misc) [inline], [explicit]

6.26.3 Member Function Documentation

6.26.3.1 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Gravity< Type, Dimension >::a (Misc &&... misc) [inline]

Access to the a data.

Provides an access to the a data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

<i>in</i>	<i>misc</i>	Miscellaneous arguments.
-----------	-------------	--------------------------

Returns

Forwarded result.

6.26.3.2 template<typename Type = double, unsigned int Dimension = 3> template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<magrathea::AbstractContents<Gravity<Type, Dimension>>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type, std::array<Type, Dimension>>>().template data<3, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type> Template Gravity< Type, Dimension >::a (Misc &&... misc) [inline]

6.26.3.3 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Gravity< Type, Dimension >::a (Misc &&... misc) const [inline]

Immutable access to the a data.

Provides an immutable access to the a data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.26.3.4 template<typename Type = double, unsigned int Dimension = 3> template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<const magrathea::AbstractContents<Gravity<Type, Dimension>>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type, std::array<Type, Dimension>>>().template data<3, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type> Template Gravity< Type, Dimension >::a (Misc &&... misc) const [inline]

6.26.3.5 template<typename Type = double, unsigned int Dimension = 3> template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<magrathea::AbstractContents<Gravity<Type, Dimension>>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type, std::array<Type, Dimension>>>().template data<4, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type> Template Gravity< Type, Dimension >::dphidt (Misc &&... misc) [inline]

6.26.3.6 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Gravity< Type, Dimension >::dphidt (Misc &&... misc) [inline]

Access to the dphidt data.

Provides an access to the dphidt data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.26.3.7 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Gravity< Type, Dimension >::dphidt (Misc &&... misc) const [inline]

Immutable access to the dphidt data.

Provides an immutable access to the dphidt data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.26.3.8 template<typename Type = double, unsigned int Dimension = 3> template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<const magrathea::AbstractContents<Gravity<Type, Dimension>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type, std::array<Type, Dimension>>>().template data<4, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type> Template Gravity< Type, Dimension >::dphidt (Misc &&... misc) const [inline]

6.26.3.9 template<typename Type = double, unsigned int Dimension = 3> template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<magrathea::AbstractContents<Gravity<Type, Dimension>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type, std::array<Type, Dimension>>>().template data<2, Values...>(0, std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type> Template Gravity< Type, Dimension >::dphidx (Misc &&... misc) [inline]

6.26.3.10 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Gravity< Type, Dimension >::dphidx (Misc &&... misc) [inline]

Access to the dphidx data.

Provides an access to the dphidx data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--

Returns

Forwarded result.

6.26.3.11 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > [Template Gravity< Type, Dimension >](#)::dphidx ([Misc &&... misc](#)) const [inline]

Immutable access to the dphidx data.

Provides an immutable access to the dphidx data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--

Returns

Forwarded result.

6.26.3.12 template<typename Type = double, unsigned int Dimension = 3> template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<const magrathea::AbstractContents<Gravity<Type, Dimension>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type, std::array<Type, Dimension>>>().template data<2, Values...>(0, std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type> [Template Gravity< Type, Dimension >](#)::dphidx ([Misc &&... misc](#)) const [inline]

6.26.3.13 template<typename Type = double, unsigned int Dimension = 3> template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<magrathea::AbstractContents<Gravity<Type, Dimension>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type, std::array<Type, Dimension>>>().template data<2, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type> [Template Gravity< Type, Dimension >](#)::dphidxyz ([Misc &&... misc](#)) [inline]

6.26.3.14 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > [Template Gravity< Type, Dimension >](#)::dphidxyz ([Misc &&... misc](#)) [inline]

Access to the dphidxyz data.

Provides an access to the dphidxyz data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

- 6.26.3.15 template<typename Type = double, unsigned int Dimension = 3> template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<const magrathea::AbstractContents<Gravity<Type, Dimension>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type, std::array<Type, Dimension>>>().template data<2, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type> Template Gravity< Type, Dimension >::dphidxyz (Misc &&... misc) const [inline]
- 6.26.3.16 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Gravity< Type, Dimension >::dphidxyz (Misc &&... misc) const [inline]

Immutable access to the dphidxyz data.

Provides an immutable access to the dphidxyz data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

- 6.26.3.17 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Gravity< Type, Dimension >::dphidy (Misc &&... misc) [inline]

Access to the dphidy data.

Provides an access to the dphidy data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.26.3.18 template<typename Type = double, unsigned int Dimension = 3> template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<magrathea::AbstractContents<Gravity<Type, Dimension>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type, std::array<Type, Dimension>>>().template data<2, Values...>(1, std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type> Template Gravity< Type, Dimension >::dphidy (Misc &&... misc) [inline]

6.26.3.19 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Gravity< Type, Dimension >::dphidy (Misc &&... misc) const [inline]

Immutable access to the dphidy data.

Provides an immutable access to the dphidy data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.26.3.20 template<typename Type = double, unsigned int Dimension = 3> template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<const magrathea::AbstractContents<Gravity<Type, Dimension>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type, std::array<Type, Dimension>>>().template data<2, Values...>(1, std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type> Template Gravity< Type, Dimension >::dphidy (Misc &&... misc) const [inline]

6.26.3.21 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Gravity< Type, Dimension >::dphidz (Misc &&... misc) [inline]

Access to the dphidz data.

Provides an access to the dphidz data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.26.3.22 template<typename Type = double, unsigned int Dimension = 3> template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<magrathea::AbstractContents<Gravity<Type, Dimension>>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type, std::array<Type, Dimension>>>>().template data<2, Values...>(2, std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type> Template Gravity< Type, Dimension >::dphidz (Misc &&... misc) [inline]

6.26.3.23 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Gravity< Type, Dimension >::dphidz (Misc &&... misc) const [inline]

Immutable access to the dphidz data.

Provides an immutable access to the dphidz data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.26.3.24 template<typename Type = double, unsigned int Dimension = 3> template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<const magrathea::AbstractContents<Gravity<Type, Dimension>>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type, std::array<Type, Dimension>>>>().template data<2, Values...>(2, std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type> Template Gravity< Type, Dimension >::dphidz (Misc &&... misc) const [inline]

6.26.3.25 template<typename Type , unsigned int Dimension> int Gravity< Type, Dimension >::example () [static]

Example function.

Tests and demonstrates the use of [Gravity](#).

Returns

0 if no error.

6.26.3.26 template<typename Type = double, unsigned int Dimension = 3> static int Gravity< Type, Dimension >::example () [static]

6.26.3.27 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Gravity< Type, Dimension >::phi (Misc &&... misc) [inline]

Access to the phi data.

Provides an access to the phi data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

- 6.26.3.28 template<typename Type = double, unsigned int Dimension = 3> template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<magrathea::AbstractContents<Gravity<Type, Dimension>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type, std::array<Type, Dimension>>>().template data<1, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type> Template Gravity< Type, Dimension >::phi (Misc &&... misc) [inline]
- 6.26.3.29 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Gravity< Type, Dimension >::phi (Misc &&... misc) const [inline]

Immutable access to the phi data.

Provides an immutable access to the phi data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

- 6.26.3.30 template<typename Type = double, unsigned int Dimension = 3> template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<const magrathea::AbstractContents<Gravity<Type, Dimension>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type, std::array<Type, Dimension>>>().template data<1, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type> Template Gravity< Type, Dimension >::phi (Misc &&... misc) const [inline]
- 6.26.3.31 template<typename Type = double, unsigned int Dimension = 3> template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<magrathea::AbstractContents<Gravity<Type, Dimension>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type, std::array<Type, Dimension>>>().template data<0, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type> Template Gravity< Type, Dimension >::rho (Misc &&... misc) [inline]
- 6.26.3.32 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Gravity< Type, Dimension >::rho (Misc &&... misc) [inline]

Access to the rho data.

Provides an access to the rho data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.26.3.33 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Gravity< Type, Dimension >::rho (Misc &&... misc) const [inline]

Immutable access to the rho data.

Provides an immutable access to the rho data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.26.3.34 template<typename Type = double, unsigned int Dimension = 3> template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<const magrathea::AbstractContents<Gravity<Type, Dimension>, magrathea::EulerianCategory, Type, Type, std::array<Type, Dimension>, Type, Type, std::array<Type, Dimension>>>().template data<0, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type> Template Gravity< Type, Dimension >::rho (Misc &&... misc) const [inline]

6.26.3.35 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Gravity< Type, Dimension >::vx (Misc &&... misc) [inline]

Access to the vx data.

Provides an access to the vx data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.26.3.36 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Gravity< Type, Dimension >::vx (Misc &&... misc) const [inline]

Immutable access to the vx data.

Provides an immutable access to the vx data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.26.3.37 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Gravity< Type, Dimension >::vxyz (Misc &&... misc) [inline]

Access to the vxyz data.

Provides an access to the vxyz data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.26.3.38 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Gravity< Type, Dimension >::vxyz (Misc &&... misc) const [inline]

Immutable access to the vxyz data.

Provides an immutable access to the vxyz data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.26.3.39 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Gravity< Type, Dimension >::vy (Misc &&... misc) [inline]

Access to the vy data.

Provides an access to the vy data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.26.3.40 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Gravity< Type, Dimension >::vy (Misc &&... misc) const [inline]

Immutable access to the vy data.

Provides an immutable access to the vy data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.26.3.41 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > **Template Gravity**<Type, Dimension >::vz(Misc &&... misc) [inline]

Access to the vz data.

Provides an access to the vz data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.26.3.42 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > **Template Gravity**<Type, Dimension >::vz(Misc &&... misc) const [inline]

Immutable access to the vz data.

Provides an immutable access to the vz data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.26.4 Member Data Documentation

6.26.4.1 template<typename Type = double, unsigned int Dimension = 3> **typedef Gravity**<Type, Dimension >::operator

The documentation for this exception was generated from the following files:

- /data/home/mbreton/magrathea_pathfinder/src/gravity.h
- /data/home/mbreton/magrathea_pathfinder/src/gravity2.h

6.27 magrathea::GridCategory Exception Reference

Category concept of grid : data related to the mesh.

```
#include <gridcategory.h>
```

Static Public Member Functions

Test

- static int [example \(\)](#)

Example function.

6.27.1 Detailed Description

Category concept of grid : data related to the mesh.

Categorizes data as a grid one. A grid data is a data involved in the navigation process through the mesh applied to a numerical simulation.

6.27.2 Member Function Documentation

6.27.2.1 int magrathea::GridCategory::example () [static]

Example function.

Tests and demonstrates the use of [GridCategory](#).

Returns

0 if no error.

The documentation for this exception was generated from the following file:

- /data/home/mbreton/magrathea_pathfinder/src/magrathea/[gridcategory.h](#)

6.28 Hmaps Class Reference

```
#include <hmmaps.h>
```

Static Public Member Functions

- template<class Parameters , class Map >
static void [ReadParamFile](#) (Parameters &[parameters](#), Map &[parameter](#))
Read parameter file.
- template<class Parameter , typename Integer , class Cones >
static void [getPixels_per_cone](#) (const Parameter &[parameters](#), const Integer npix, std::vector< Integer > &[pixel](#), std::vector< unsigned int > &[ntrajectories](#), const unsigned int iconerank, const Cones &[cones](#))
Get targets.
- template<class Parameter , typename Integer , class Cones >
static void [getPixels_per_cone2](#) (const Parameter &[parameters](#), const Integer npix, std::vector< Integer > &[pixel](#), std::vector< unsigned int > &[ntrajectories](#), const unsigned int iconerank, const Cones &[cones](#))
Get targets.
- template<class Parameter , class Octree , class Map , class Pixel , class Cosmology , class Point , typename Integer , typename Real >
static void [FillMapPropagate](#) (const Parameter &[parameters](#), const Integer ntrajectory, const Integer firsttrajectory, const Octree &[octree](#), const Point &[vobs](#), Map &[map](#), const Integer nmaps, const Pixel &[pixel](#), const Cosmology &[cosmology](#), const point &[observer](#), const Real length, const std::vector< Real > &[z_stop_vec](#))
Produce Healpix maps using bundles.

- template<class Parameter , class Octree , class Map , class Pixel , class Cosmology , class Point , typename Integer , typename Real >
 static void **FillMap** (const Parameter &**parameters**, const std::vector< std::string > &**map_components**, const std::vector< unsigned int > &**index_components**, const Integer **ntrajectory**, const Integer **firsttrajectory**, const Octree &**octree**, const Point &**vobs**, Map &**map**, const unsigned int **nmaps**, const Pixel &**pixel**, const Cosmology &**cosmology**, const **point** &**observer**, const Real **length**, const std::vector< Real > &**interpRefvec**, const std::vector< Real > &**rhomo**, const std::vector< Real > &**thomo**, const std::vector< Real > &**lambdahomo**, const std::vector< Real > &**redshifthomo**, const std::vector< Real > &**ahomo**)

Produce Healpix maps.

6.28.1 Member Function Documentation

- 6.28.1.1 template<class Parameter , class Octree , class Map , class Pixel , class Cosmology , class Point , typename Integer , typename Real > void Hmaps::FillMap (const Parameter & **parameters**, const std::vector< std::string > & **map_components**, const std::vector< unsigned int > & **index_components**, const Integer **ntrajectory**, const Integer **firsttrajectory**, const Octree & **octree**, const Point & **vobs**, Map & **map**, const unsigned int **nmaps**, const Pixel & **pixel**, const Cosmology & **cosmology**, const point & **observer**, const Real **length**, const std::vector< Real > & **interpRefvec**, const std::vector< Real > & **rhomo**, const std::vector< Real > & **thomo**, const std::vector< Real > & **lambdahomo**, const std::vector< Real > & **redshifthomo**, const std::vector< Real > & **ahomo**) [static]

Produce Healpix maps.

Produce Healpix maps

Template Parameters

<i>Parameter</i>	Parameter type
<i>Octree</i>	octree type
<i>Map</i>	map type
<i>Pixel</i>	pixel type
<i>Cosmology</i>	cosmology type
<i>Point</i>	Point type
<i>Integer</i>	integer type
<i>Real</i>	real/float type

Parameters

in	<i>parameters</i>	Parameter structure.
in	<i>map_- components</i>	Vector of strings containing the map types.
in	<i>index_- components</i>	Vector containing the index relative to the map types.
in	<i>ntrajectory</i>	Number of pixels to be filled
in	<i>firsttrajectory</i>	first index for pixel for a given cone
in	<i>octree</i>	Octree
in	<i>vobs</i>	Observer velocity
in,out	<i>map</i>	Map
in	<i>nmaps</i>	Number of Healpix maps
in	<i>pixel</i>	Pixel vector
in	<i>cosmology</i>	Cosmological table
in	<i>observer</i>	Observer position
in	<i>length</i>	R.U to S.I units for length
in	<i>interpRefvec</i>	vector of interpolations at a given quantity for a given surface
in	<i>rhomo</i>	vector of FLRW distance at the redshifts of interest
in	<i>thomo</i>	vector of FLRW time at the redshifts of interest
in	<i>lambdahomo</i>	vector of FLRW affine parameter at the redshifts of interest
in	<i>redshifthomo</i>	vector of FLRW redshift at the redshifts of interest
in	<i>ahomo</i>	vector of FLRW scale factor at the redshifts of interest

6.28.1.2 template<class Parameter , class Octree , class Map , class Pixel , class Cosmology , class Point , typename Integer , typename Real > void Hmaps::FillMapPropagate (const Parameter & *parameters*, const Integer *ntrajectory*, const Integer *firsttrajectory*, const Octree & *octree*, const Point & *vobs*, Map & *map*, const Integer *nmaps*, const Pixel & *pixel*, const Cosmology & *cosmology*, const point & *observer*, const Real *length*, const std::vector< Real > & *z_stop_vec*) [static]

Produce Healpix maps using bundles.

Produce Healpix maps using bundles at constant FLRW redshift

Template Parameters

<i>Parameter</i>	Parameter type
<i>Octree</i>	octree type
<i>Map</i>	map type
<i>Pixel</i>	pixel type
<i>Cosmology</i>	cosmology type
<i>Integer</i>	integer type
<i>Real</i>	real/float type

Parameters

in	<i>parameters</i>	Parameter structure
in	<i>ntrajectory</i>	Number of pixels to be filled
in	<i>firsttrajectory</i>	first index for pixel for a given cone
in	<i>octree</i>	Octree
in	<i>vobs</i>	Observer velocity
in,out	<i>map</i>	Map
in	<i>nmaps</i>	Number of Healpix maps
in	<i>pixel</i>	Pixel vector
in	<i>cosmology</i>	Cosmological table
in	<i>observer</i>	Observer position
in	<i>length</i>	R.U to S.I units for length
in	<i>z_stop_vec</i>	Vector containing redshifts at which we compute scalar quantities

6.28.1.3 template<class Parameter , typename Integer , class Cones > void Hmaps::getPixels_per_cone (const Parameter & *parameters*, const Integer *npix*, std::vector< Integer > & *pixel*, std::vector< unsigned int > & *ntrajectories*, const unsigned int *iconerank*, const Cones & *cones*) [static]

Get targets.

Get targets inside cone.

Template Parameters

<i>Parameter</i>	Parameter type
<i>Integer</i>	Integer type
<i>Cones</i>	Cones type

Parameters

in	<i>parameters</i>	Parameter structure.
in	<i>npix</i>	Number of pixels.
in,out	<i>pixel</i>	Vector of pixels inside cones that are treated by the same MPI task.
in	<i>ntrajectories</i>	Number of pixels per cone.
in	<i>iconerank</i>	Number of the cone of interest.
in	<i>Cones</i>	Geometry of all the cones. return Filled vector of targets inside cones for a given MPI task

6.28.1.4 template<class Parameter , typename Integer , class Cones > void Hmaps::getPixels_per_cone2 (const Parameter & parameters, const Integer npix, std::vector< Integer > & pixel, std::vector< unsigned int > & ntrajectories, const unsigned int iconerank, const Cones & cones) [static]

Get targets.

Get targets inside cone.

Template Parameters

<i>Parameter</i>	Parameter type
<i>Integer</i>	Integer type
<i>Cones</i>	Cones type

Parameters

in	<i>parameters</i>	Parameter structure.
in	<i>npix</i>	Number of pixels.
in,out	<i>pixel</i>	Vector of pixels inside cones that are treated by the same MPI task.
in	<i>ntrajectories</i>	Number of pixels per cone.
in	<i>iconerank</i>	Number of the cone of interest.
in	<i>Cones</i>	Geometry of all the cones. return Filled vector of targets inside cones for a given MPI task

6.28.1.5 template<class Parameters , class Map > void Hmaps::ReadParamFile (Parameters & parameters, Map & parameter) [static]

Read parameter file.

Read and put in a structure the parameters.

Template Parameters

<i>Parameters</i>	structure type
<i>Map</i>	map type

Parameters

in,out	<i>parameters</i>	Structure containing the parameters.
in	<i>parameter</i>	Contains parameters to be rewritten

Returns

Filled parameters structure.

The documentation for this class was generated from the following file:

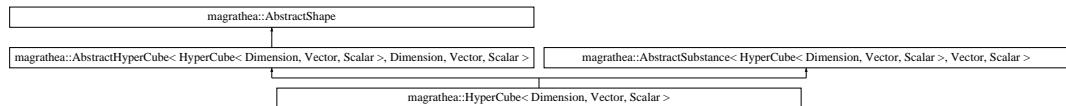
- /data/home/mbreton/magrathea_pathfinder/src/[hmaps.h](#)

6.29 magrathea::HyperCube< Dimension, Vector, Scalar > Exception Template Reference

N-dimensional cube.

```
#include <hypercube.h>
```

Inheritance diagram for magrathea::HyperCube< Dimension, Vector, Scalar >:



Public Member Functions

Lifecycle

- template<class... Misc>
HyperCube (Misc &&...misc)
Explicit generic constructor.

Data

- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<AbstractSubstance<HyperCube<-Dimension, Vector, Scalar>, Vector, Scalar> >().template data<0, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type>
Template position (Misc &&...misc)
Access to the position data.
- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<const AbstractSubstance<HyperCube<-Dimension, Vector, Scalar>, Vector, Scalar> >().template data<0, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type>
Template position (Misc &&...misc) const
Immutable access to the position data.
- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<AbstractSubstance<HyperCube<-Dimension, Vector, Scalar>, Vector, Scalar> >().template data<1, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type>
Template extent (Misc &&...misc)
Access to the extent data.
- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<const AbstractSubstance<HyperCube<-Dimension, Vector, Scalar>, Vector, Scalar> >().template data<1, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type>
Template extent (Misc &&...misc) const
Immutable access to the extent data.

Static Public Member Functions

Predefined

- static constexpr **HyperCube**
< Dimension, Vector, Scalar > **unit** ()
Unit hypercube.

Test

- static int **example** ()
Example function.

Public Attributes

- using **operator** = **typedef**

Additional Inherited Members

6.29.1 Detailed Description

```
template<unsigned int Dimension = 3, class Vector = std::array<double, Dimension>, typename Scalar = typename std::remove_cv<typename std::remove_reference<decltype(std::declval<Vector>()[0])>::type>::type>exception magrathea::HyperCube<Dimension, Vector, Scalar >
```

N-dimensional cube.

Implementation of a basic hypercube of arbitrary dimension.

Template Parameters

<i>Dimension</i>	Number of space dimension.
<i>Vector</i>	Position vector type.
<i>Scalar</i>	Scalar data type.

6.29.2 Constructor & Destructor Documentation

6.29.2.1 template<unsigned int Dimension, class Vector , typename Scalar > template<class... Misc> magrathea::HyperCube< Dimension, Vector, Scalar >::HyperCube (*Misc* &&... *misc*) [inline], [explicit]

Explicit generic constructor.

Provides a generic interface to all constructors of the base class.

Template Parameters

<i>Misc</i>	(Miscellaneous types.)
-------------	------------------------

Parameters

<i>in</i>	<i>misc</i>	Miscellaneous arguments.
-----------	-------------	--------------------------

6.29.3 Member Function Documentation

6.29.3.1 template<unsigned int Dimension, class Vector , typename Scalar > int magrathea::HyperCube< Dimension, Vector, Scalar >::example () [static]

Example function.

Tests and demonstrates the use of [HyperCube](#).

Returns

0 if no error.

6.29.3.2 template<unsigned int Dimension, class Vector , typename Scalar > template<unsigned int... Values, class... Misc, class Template , class > Template magrathea::HyperCube< Dimension, Vector, Scalar >::extent (*Misc* &&... *misc*) [inline]

Access to the extent data.

Provides an access to the extent data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.29.3.3 template<unsigned int Dimension, class Vector , typename Scalar > template<unsigned int... Values, class... Misc, class Template , class > Template magrathea::HyperCube< Dimension, Vector, Scalar >::extent (Misc &&... misc) const [inline]

Immutable access to the extent data.

Provides an immutable access to the extent data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.29.3.4 template<unsigned int Dimension, class Vector , typename Scalar > template<unsigned int... Values, class... Misc, class Template , class > Template magrathea::HyperCube< Dimension, Vector, Scalar >::position (Misc &&... misc) [inline]

Access to the position data.

Provides an access to the position data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.29.3.5 template<unsigned int Dimension, class Vector , typename Scalar > template<unsigned int... Values, class... Misc, class Template , class > Template magrathea::HyperCube< Dimension, Vector, Scalar >::position (Misc &&... misc) const [inline]

Immutable access to the position data.

Provides an immutable access to the position data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.29.3.6 template<unsigned int Dimension, class Vector , typename Scalar > constexpr HyperCube< Dimension, Vector, Scalar > magrathea::HyperCube< Dimension, Vector, Scalar >::unit () [static]

Unit hypercube.

Creates an hypercube with a position of zero and with an extent of one.

Returns

Copy of a unit hypercube.

6.29.4 Member Data Documentation

6.29.4.1 template<unsigned int Dimension = 3, class Vector = std::array<double, Dimension>, typename Scalar = typename std::remove_cv<typename std::remove_reference<decltype(std::declval<Vector>()[0])>::type>::type> using magrathea::HyperCube< Dimension, Vector, Scalar >::operator =

The documentation for this exception was generated from the following file:

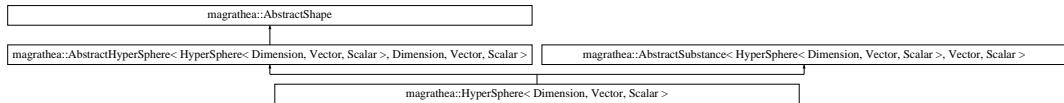
- /data/home/mbreton/magrathea_pathfinder/src/magrathea/[hypercube.h](#)

6.30 magrathea::HyperSphere< Dimension, Vector, Scalar > Exception Template Reference

N-dimensional sphere.

```
#include <hypersphere.h>
```

Inheritance diagram for magrathea::HyperSphere< Dimension, Vector, Scalar >:



Public Member Functions

Lifecycle

- template<class... Misc>
HyperSphere (Misc &&...misc)
Explicit generic constructor.

Data

- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<AbstractSubstance<HyperSphere<-Dimension, Vector, Scalar>, Vector, Scalar> >().template data<0, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type>
Template position (Misc &&...misc)
Access to the position data.
- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<const AbstractSubstance<HyperSphere<-Dimension, Vector, Scalar>, Vector, Scalar> >().template data<0, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type>
Template position (Misc &&...misc) const
Immutable access to the position data.
- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<AbstractSubstance<HyperSphere<-Dimension, Vector, Scalar>, Vector, Scalar> >().template data<1, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type>
Template extent (Misc &&...misc)
Access to the extent data.
- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<const AbstractSubstance<HyperSphere<-Dimension, Vector, Scalar>, Vector, Scalar> >().template data<1, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type>
Template extent (Misc &&...misc) const
Immutable access to the extent data.

Static Public Member Functions

Predefined

- static constexpr **HyperSphere**
< Dimension, Vector, Scalar > **unit** ()
Unit hypersphere.

Test

- static int **example** ()
Example function.

Public Attributes

- using **operator** = **typedef**

Additional Inherited Members

6.30.1 Detailed Description

```
template<unsigned int Dimension = 3, class Vector = std::array<double, Dimension>, typename Scalar = typename std::remove_cv<typename std::remove_reference<decltype(std::declval<Vector>()[0])>::type>::type>exception magrathea::HyperSphere< Dimension, Vector, Scalar >
```

N-dimensional sphere.

Implementation of a basic hypersphere of arbitrary dimension.

Template Parameters

<i>Dimension</i>	Number of space dimension.
<i>Vector</i>	Position vector type.
<i>Scalar</i>	Scalar data type.

6.30.2 Constructor & Destructor Documentation

6.30.2.1 template<unsigned int Dimension, class Vector , typename Scalar > template<class... Misc> magrathea::HyperSphere< Dimension, Vector, Scalar >::HyperSphere (*Misc* &&... *misc*) [inline], [explicit]

Explicit generic constructor.

Provides a generic interface to all constructors of the base class.

Template Parameters

<i>Misc</i>	(Miscellaneous types.)
-------------	------------------------

Parameters

<i>in</i>	<i>misc</i>	Miscellaneous arguments.
-----------	-------------	--------------------------

6.30.3 Member Function Documentation

6.30.3.1 template<unsigned int Dimension, class Vector , typename Scalar > int magrathea::HyperSphere< Dimension, Vector, Scalar >::example () [static]

Example function.

Tests and demonstrates the use of [HyperSphere](#).

Returns

0 if no error.

6.30.3.2 template<unsigned int Dimension, class Vector , typename Scalar > template<unsigned int... Values, class... Misc, class Template , class > Template magrathea::HyperSphere< Dimension, Vector, Scalar >::extent (*Misc* &&... *misc*) [inline]

Access to the extent data.

Provides an access to the extent data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.30.3.3 template<unsigned int Dimension, class Vector , typename Scalar > template<unsigned int... Values, class... Misc, class Template , class > Template magrathea::HyperSphere< Dimension, Vector, Scalar >::extent (Misc &&... misc) const [inline]

Immutable access to the extent data.

Provides an immutable access to the extent data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.30.3.4 template<unsigned int Dimension, class Vector , typename Scalar > template<unsigned int... Values, class... Misc, class Template , class > Template magrathea::HyperSphere< Dimension, Vector, Scalar >::position (Misc &&... misc) [inline]

Access to the position data.

Provides an access to the position data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.30.3.5 template<unsigned int Dimension, class Vector , typename Scalar > template<unsigned int... Values, class... Misc, class Template , class > Template magrathea::HyperSphere< Dimension, Vector, Scalar >::position (Misc &&... misc) const [inline]

Immutable access to the position data.

Provides an immutable access to the position data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.30.3.6 template<unsigned int Dimension, class Vector , typename Scalar > constexpr HyperSphere< Dimension, Vector, Scalar > magrathea::HyperSphere< Dimension, Vector, Scalar >::unit() [static]

Unit hypersphere.

Creates an hypersphere with a position of zero and with an extent of one.

Returns

Copy of a unit hypersphere.

6.30.4 Member Data Documentation

6.30.4.1 template<unsigned int Dimension = 3, class Vector = std::array<double, Dimension>, typename Scalar = typename std::remove_cv<typename std::remove_reference<decltype(std::declval<Vector>()[0])>::type>::type> using magrathea::HyperSphere< Dimension, Vector, Scalar >::operator =

The documentation for this exception was generated from the following file:

- /data/home/mbreton/magrathea_pathfinder/src/magrathea/[hypersphere.h](#)

6.31 Input Exception Reference

[Input](#) utilities for raytracing.

```
#include <input.h>
```

Static Public Member Functions

Utilities

- static std::string **trim** (const std::string &text, const std::string &comment="#")

Trim string.
- static std::pair< std::string, std::string > **partition** (const std::string &text, const std::string &separator="=")

Partition string.
- template<class Octree , class Source , class Data = typename std::tuple_element<1, decltype(Octree::element())>::type, class Element = decltype(Source::element()), class = typename std::enable_if<(Octree::dimension() == Source::dimension()) && (std::is_integral<Data>::value)>::type>
 static unsigned int **count** (Octree &octree, const Source &source)

Count tree.
- template<class Octree , class Sphere , class Conic , typename Type = decltype(Octree::type()), class Element = decltype(Octree::element()), class Index = typename std::tuple_element<0, Element>::type, unsigned int Dimension = Octree::dimension(), class Position = decltype(Octree::position()), class Extent = decltype(Octree::extent()), class = typename std::enable_if<(Dimension == 3) && (Dimension == Sphere::dimension())>::type>
 static bool **collide** (const Octree &octree, const Index &index, const Sphere &sphere, const Conic &conic)

Collision between an octree index and a sphere or a cone.
- template<unsigned int Selection = 0, class Octree , unsigned int Dimension = Octree::dimension(), class Element = decltype(Octree::element()), class Index = typename std::tuple_element<0, Element>::type, class Data = typename std::tuple_element<1, Element>::type, class Type = decltype(Data::template type<Selection>()), class = typename std::enable_if<(Dimension == 3)>::type>
 static Type **mean** (const Octree &octree, const Element &element, int level=-1)

Mean value over cells.
- template<unsigned int Selection = 0, class Octree , unsigned int Dimension = Octree::dimension(), class Element = decltype(Octree::element()), class Index = typename std::tuple_element<0, Element>::type, class Data = typename std::tuple_element<1, Element>::type, class Type = decltype(Data::template type<Selection>()), class = typename std::enable_if<(Dimension == 3)>::type>
 static Data **meanAll** (const Octree &octree, const Element &element, int level=-1)

Mean value over cells.
- template<typename Type , class Cosmology = std::array<std::vector<double>, 4>, class = typename std::enable_if<std::is_convertible<Type, typename std::remove_cv<typename std::remove_reference<decltype(std::declval<Cosmology>()[0][0])>::type>::value>::type>
 static Cosmology **constantify** (const unsigned int size, const Type tmin, const Type tmax, const Type a=Type(1), const Type dadt=Type(), const Type d2adt2=Type())

Constant cosmology.
- template<class Parameter , class Data , typename Type , class = typename std::enable_if<Data::types() != 0>::type>
 static Data **sistemize** (const Parameter &**parameters**, const Data &data, const Type a, const Type h, const Type omegam, const Type lboxmpch)

Data conversion to SI units.
- template<class Parameter , class Octree , typename Type , class Element = decltype(Octree::element()), class Data = typename std::tuple_element<1, Element>::type, class = typename std::enable_if<(!std::is_void<Data>::value) && (Octree::dimension() != 0)>::type>
 static unsigned int **sistemize** (const Parameter &**parameters**, Octree &octree, const Type h, const Type omegam, const Type lboxmpch)

Octree conversion to SI units.
- template<class Data , class = typename std::enable_if<Data::types() != 0>::type>
 static Data **homogenize** (const Data &data)

Data homogenization.
- template<class Octree , class... Dummy, class Element = decltype(Octree::element()), class Data = typename std::tuple_element<1, Element>::type, class = typename std::enable_if<(!std::is_void<Data>::value) && (Octree::dimension() != 0) && (sizeof...(Dummy) == 0)>::type>
 static unsigned int **homogenize** (Octree &octree, Dummy...)

Octree homogenization.
- template<class Extent = std::ratio<1>, class Data , class Vector , typename Type , unsigned int Dimension = 3, class = typename std::enable_if<(<Data::types() != 0) && (std::is_convertible<Type, typename std::remove_cv<typename std::remove_reference<decltype(std::declval<Vector>()[0])>::type>::value)>::type>

static Data [schwarzschildify](#) (const Data &data, const Vector ¢er, const Vector &position, const Type mass, const Type length)

Data schwarzschild setup.

- template<class Octree , class Vector , typename Type , class Function , class... Dummy, class Element = decltype(Octree::element()), unsigned int Dimension = Octree::dimension(), class Extent = decltype(Octree::extent()), class = typename std::enable_if<(sizeof...(Dummy) == 0) && (Dimension == 3)>::type>

static unsigned int [schwarzschildify](#) (Octree &octree, const Vector &position, const Type mass, const Type length, Function &&refiner, Dummy...)

Octree schwarzschild setup.

Files

- template<class Octree , class Element = decltype(Octree::element()), class Index = typename std::tuple_element<0, Element>::type, class Data = typename std::tuple_element<1, Element>::type, unsigned int Dimension = Octree::dimension(), class = typename std::enable_if<Dimension != 0>::type>

static unsigned int [filetree](#) (Octree &octree, const std::string &directory, const std::string &format)

File tree.

- template<class List , class Octree , class Sphere , class Conic , unsigned int Dimension = Octree::dimension(), class = typename std::enable_if<(Dimension == 3) && (Dimension == Sphere::dimension())>::type>

static bool [prepare](#) (List &list, const Octree &octree, const Sphere &sphere, const Conic &conic)

File list preparation.

Data

- template<typename Integral = unsigned int, typename Real = float, class Parameter , class Octree , class Function , class Element = decltype(Octree::element()), class Index = typename std::tuple_element<0, Element>::type, class Data = typename std::tuple_element<1, Element>::type, unsigned int Dimension = Octree::dimension(), class = typename std::enable_if<std::is_convertible<typename std::result_of<Function(Element)>::type, bool>::value>::type>

static bool [import](#) (const Parameter &[parameters](#), Octree &octree, const std::string &filename, Function &&filter)

Ramses importation.

- template<typename Type , class Parameter , class Cosmology = std::array<std::vector<Type>, 4>, class Element = typename std::remove_cv<typename std::remove_reference<decltype(std::declval<Cosmology>()[0])>::type>::type, class = typename std::enable_if<std::is_convertible<Element, typename std::remove_cv<typename std::remove_reference<decltype(std::declval<Cosmology>()[0])>::type>::value>::type>

static Cosmology [acquire](#) (const Parameter &[parameters](#), Type &h, Type &omegam, Type &lboxmpch, const std::string &outfile=std::string())

Cosmology acquisition.

- template<class Container = std::map<std::string, std::string>, class Element = std::pair<std::string, std::string>, class = typename std::enable_if<(std::is_convertible<Container, std::map<std::string, std::string> >::value) && (std::is_convertible<Element, std::pair<std::string, std::string> >::value)>::type>

static Container [parse](#) (const std::string &filename, const std::string &separator="=", const std::string &comment="#")

Parameter file parsing.

- template<class Position , class Extent , typename Integral = unsigned int, typename Real = float, class Parameter , class Conic , class Octree , class Function , class Element = decltype(Octree::element()), class Index = typename std::tuple_element<0, Element>::type, class Data = typename std::tuple_element<1, Element>::type, unsigned int Dimension = Octree::dimension(), class = typename std::enable_if<std::is_convertible<typename std::result_of<Function(Element)>::type, bool>::value>::type>

static bool [import hdf5](#) (const Parameter &[parameters](#), unsigned int &rank, const std::array< std::array< double, 3 >, 3 > &rotm1, const double &thetay, const double &thetaz, const Conic &conic, Octree &octree, const std::string &filename, Function &&filter)

Ramses importation.

- template<class Position , class Extent , typename Integral = unsigned int, typename Real = float, class Parameter , class Octree , class Function , class Element = decltype(Octree::element()), class Index = typename std::tuple_element<0, Element>::type, class Data = typename std::tuple_element<1, Element>::type, unsigned int Dimension = Octree::dimension(), class = typename std::enable_if<std::is_convertible<typename std::result_of<Function(Element)>::type, bool>::value>::type>

static bool [importfullhdf5](#) (const Parameter &[parameters](#), Octree &octree, const std::string &filename, Function &&filter)

Ramsey importation.

- template<class Position , class Extent , typename Integral = unsigned int, typename Real = float, class Parameter , class Octree , class Function , class Element = decltype(Octree::element()), class Index = typename std::tuple_element<0, Element>::type, class Data = typename std::tuple_element<1, Element>::type, unsigned int Dimension = Octree::dimension(), class = typename std::enable_if<std::is_convertible<typename std::result_of<Function(Element)>::type, bool>::value>::type>
 static bool **importascii** (const Parameter &**parameters**, const std::array< std::array< double, 3 >, 3 > &rotm1, Octree &octree, const std::string &filename, Function &&filter)

Ramsey importation.

Cones

- template<class Octree , class = typename std::enable_if<Octree::dimension() != 0>::type>
 static bool **save** (Octree &octree, const std::string &filename)

Save temporary cone file.
- template<class Octree , class = typename std::enable_if<Octree::dimension() != 0>::type>
 static bool **load** (Octree &octree, const std::string &filename)

Load temporary cone file.

Correction

- template<class Cosmology , class Trajectory , class Type = typename std::remove_cv<typename std::remove_reference<decltype(std::declval<typename std::tuple_element<0, Cosmology>::type>()[0])>::type>::type, class = typename std::enable_if<std::is_convertible<Type, typename std::remove_cv<typename std::remove_reference<decltype(std::declval<typename std::tuple_element<0, Cosmology>::type>()[0])>::type>::value>::type>
 static Cosmology **correct** (const Cosmology &cosmology, const Trajectory &trajectory)

Cosmology correction.
- template<class Parameter , int Check = 0, unsigned int Selection = Check*(Check >= 0), class Octree , typename Kind = double, unsigned int Dimension = Octree::dimension(), class Element = decltype(Octree::element()), class Index = typename std::tuple_element<0, Element>::type, class Data = typename std::tuple_element<1, Element>::type, class Type = decltype(Data::template type<Selection>()), class Position = decltype(Octree::position()), class Extent = decltype(Octree::extent()), class = typename std::enable_if<(Dimension == 3)>::type>
 static Octree & **correct** (const Parameter &**parameters**, Octree &octree, Kind &&amin=Kind())

Octree correction.

6.31.1 Detailed Description

Input utilities for raytracing.

Provides a list of importation routines to load data for raytracing.

6.31.2 Member Function Documentation

6.31.2.1 template<typename Type , class Parameter , class Cosmology , class Element , class > Cosmology **Input::acquire** (const Parameter & **parameters**, Type & h, Type & omegam, Type & lboxmpch, const std::string & **outfile** = std::string()) [static]

Cosmology acquisition.

Acquires cosmology parameters and cosmology from input files: the Hubble parameter, the density parameter of matter and the size of the box. It returns a container with t, a(t), its first and second derivatives (where t can be either the cosmic time or the conformal time eta).

Template Parameters

Type	Arithmetic type.
Cosmology	Cosmology container type.
Element	Element type of the container.

Parameters

in	<i>simfile</i>	Name of a simulation file or directory to extract the box size.
in	<i>paramfile</i>	Name of a file of cosmological parameters.
in	<i>evolfile</i>	Name of a file of cosmological evolution.
in	<i>h</i>	Hubble parameter.
in	<i>omegam</i>	Density parameter of matter.
in	<i>lboxmpch</i>	Box size in megaparsecs times h.
in	<i>mpc</i>	Value of one megaparsec in SI units.
in	<i>outfile</i>	Name of an output file for debugging purposes.

Returns

Container of cosmology evolution and derivatives. Contains t, a, H, int Hdot, Hdot

```
6.31.2.2 template<class Octree , class Sphere , class Conic , typename Type , class Element , class Index , unsigned int Dimension, class Position , class Extent , class > bool Input::collide ( const Octree & octree, const Index & index, const Sphere & sphere, const Conic & conic ) [inline], [static]
```

Collision between an octree index and a sphere or a cone.

Detects collision between an index of an octree and a sphere or a cone.

Template Parameters

<i>Octree</i>	Octree type.
<i>Sphere</i>	Sphere type.
<i>Conic</i>	Cone type.
<i>Type</i>	Scalar position type.
<i>Element</i>	Underlying element type.
<i>Index</i>	Index type.
<i>Dimension</i>	Number of dimensions.
<i>Position</i>	Position of the hyperoctree center.
<i>Extent</i>	Extent of the hyperoctree.

Parameters

in	<i>octree</i>	Input octree.
in	<i>index</i>	Index of one element.
in	<i>sphere</i>	Geometrical sphere.
in	<i>cone</i>	Three dimensional cone.

Returns

True if collision, false otherwise.

```
6.31.2.3 template<typename Type , class Cosmology , class > Cosmology Input::constantify ( const unsigned int size, const Type tmin, const Type tmax, const Type a = Type (1) , const Type dadt = Type () , const Type d2adt2 = Type () ) [inline], [static]
```

Constant cosmology.

Produces a constant cosmology of the provided size.

Template Parameters

<i>Cosmology</i>	Cosmology container type.
<i>Type</i>	Data type.

Parameters

in	<i>size</i>	Size of the cosmology tables.
in	<i>tmin</i>	First value of time.
in	<i>tmax</i>	Last value of time.
in	<i>a</i>	Scale factor.
in	<i>dadt</i>	Value of the first derivative of the scale factor.
in	<i>d2adt2</i>	Value of the second derivative of the scale factor.

Returns

Constant cosmology.

6.31.2.4 template<class Cosmology , class Trajectory , class Type , class > Cosmology Input::correct (const Cosmology & cosmology, const Trajectory & trajectory) [static]

Cosmology correction.

Produces another cosmology based on interpolation of an homogeneous trajectory.

Template Parameters

<i>Cosmology</i>	Cosmology container type.
<i>Trajectory</i>	Trajectory container.
<i>Type</i>	Data type.

Parameters

in	<i>cosmology</i>	Cosmology.
in	<i>trajectory</i>	Homogeneous trajectory.

Returns

Corrected cosmology.

6.31.2.5 template<class Parameter , int Check, unsigned int Selection, class Octree , typename Kind , unsigned int Dimension, class Element , class Index , class Data , class Type , class Position , class Extent , class > Octree & Input::correct (const Parameter & parameters, Octree & octree, Kind && amin = Kind()) [static]

Octree correction.

Corrects uncomplete tree and zones with empty rho.

Template Parameters

<i>Parameter</i>	Parameter type.
<i>Check</i>	Check mode.
<i>Selection</i>	Index of the data to be corrected.
<i>Octree</i>	Octree type.
<i>Kind</i>	Kind of amin.
<i>Dimension</i>	Number of dimensions.
<i>Element</i>	Underlying element type.
<i>Index</i>	Index type.
<i>Data</i>	Data type.
<i>Type</i>	Selected data type.
<i>Position</i>	Position of the hyperoctree center.
<i>Extent</i>	Extent of the hyperoctree.

Parameters

in	<i>parameters</i>	Parameter structure.
in,out	<i>octree</i>	Input octree.
out	<i>amin</i>	Outputs the lowest value of a.

Returns

Reference to the octree.

6.31.2.6 `template<class Octree , class Source , class Data , class Element , class > unsigned int Input::count (Octree & octree, const Source & source) [inline], [static]`

Count tree.

Counts the number of input cells in each output cells.

Template Parameters

<i>Octree</i>	Output octree type.
<i>Source</i>	Input octree type.
<i>Data</i>	Output data type.
<i>Element</i>	Underlying input element type.

Parameters

in	<i>octree</i>	Output octree.
in	<i>source</i>	Input octree.

Returns

Octree with count of input cells per output cell.

6.31.2.7 `template<class Octree , class Element , class Index , class Data , unsigned int Dimension, class > unsigned int Input::filetree (Octree & octree, const std::string & directory, const std::string & format) [static]`

File tree.

Fills in an octree with the names of the existing ramses files.

Template Parameters

<i>Octree</i>	Octree type.
<i>Element</i>	Underlying element type.
<i>Index</i>	Index type.
<i>Data</i>	Data type.
<i>Dimension</i>	Number of dimensions.

Parameters

out	<i>octree</i>	Octree of file names.
in	<i>directory</i>	Input directory.
in	<i>format</i>	Input file format.

Returns

Number of detected files.

6.31.2.8 template<class Data , class > Data Input::homogenize(const Data & data) [inline], [static]

Data homogenization.

Converts a data to one of a homogeneous empty Universe.

Template Parameters

<i>Data</i>	Data type.
-------------	------------

Parameters

in	<i>data</i>	Input data.
----	-------------	-------------

Returns

Homogeneous empty data.

6.31.2.9 template<class Octree , class... Dummy, class Element , class Data , class > unsigned int Input::homogenize(Octree & octree, Dummy...) [inline], [static]

Octree homogenization.

Converts each data of the octree to one of a homogeneous empty Universe.

Template Parameters

<i>Octree</i>	Octree type.
<i>Dummy</i>	Dummy type.
<i>Element</i>	Underlying element type.
<i>Data</i>	Data type.

Parameters

in, out	<i>octree</i>	Octree of data.
---------	---------------	-----------------

Returns

Homogeneous empty octree.

6.31.2.10 template<typename Integral , typename Real , class Parameter , class Octree , class Function , class Element , class Index , class Data , unsigned int Dimension, class > bool Input::import(const Parameter & parameters, Octree & octree, const std::string & filename, Function && filter) [static]

Ramses importation.

Imports raw data from ramses gravity files. All cells selected by the provided filter are added to the octree. As there is no way to detect the coarse level, it should be specified as an argument.

Template Parameters

<i>Integral</i>	Integral type of the file.
<i>Real</i>	Real type of the file.
<i>Parameter</i>	Parameter type.
<i>Octree</i>	Octree type.
<i>Function</i>	Function type taking an element as argument and returning a boolean.
<i>Element</i>	Underlying element type.
<i>Index</i>	Index type.
<i>Data</i>	Data type.
<i>Dimension</i>	Number of dimensions.

Parameters

in	<i>parameters</i>	Parameter structure.
in, out	<i>octree</i>	Octree of data.
in	<i>filename</i>	Input file name.
in	<i>filter</i>	Filtering algorithm of cells.

Returns

True on success, false on error.

```
6.31.2.11 template<class Position , class Extent , typename Integral , typename Real , class Parameter , class Octree , class Function , class Element , class Index , class Data , unsigned int Dimension, class > bool Input::importascii ( const Parameter & parameters, const std::array< std::array< double, 3 >, 3 > & rotm1, Octree & octree, const std::string & filename, Function && filter ) [static]
```

Ramses importation.

Imports ASCII data from ramses gravity files. All cells selected by the provided filter are added to the octree. As there is no way to detect the coarse level, it should be specified as an argument.

Template Parameters

<i>Position</i>	Position type.
<i>Extent</i>	Extent type.
<i>Integral</i>	Integral type of the file.
<i>Real</i>	Real type of the file.
<i>Octree</i>	Octree type.
<i>Function</i>	Function type taking an element as argument and returning a boolean.
<i>Element</i>	Underlying element type.
<i>Index</i>	Index type.
<i>Data</i>	Data type.
<i>Dimension</i>	Number of dimensions.

Parameters

in	<i>parameters</i>	Parameter structure
in	<i>rotm1</i>	rotation matrix for narrow cones
in, out	<i>octree</i>	Octree of data.
in	<i>filename</i>	Input file name.
in	<i>filter</i>	Filtering algorithm of cells.

Returns

True on success, false on error.

6.31.2.12 template<class Position , class Extent , typename Integral , typename Real , class Parameter , class Octree , class Function , class Element , class Index , class Data , unsigned int Dimension, class > bool Input::importfullhdf5 (const Parameter & parameters, Octree & octree, const std::string & filename, Function && filter) [static]

Ramsey importation.

Imports HDF5 data from ramsey gravity files. Used for small spherical buffer zone in narrow cones. All cells selected by the provided filter are added to the octree.

Template Parameters

<i>Position</i>	Position type.
<i>Extent</i>	Extent type.
<i>Integral</i>	Integral type of the file.
<i>Real</i>	Real type of the file.
<i>Parameter</i>	Parameter type.
<i>Octree</i>	Octree type.
<i>Function</i>	Function type taking an element as argument and returning a boolean.
<i>Element</i>	Underlying element type.
<i>Index</i>	Index type.
<i>Data</i>	Data type.
<i>Dimension</i>	Number of dimensions.

Parameters

in	<i>parameters</i>	Parameter structure.
in, out	<i>octree</i>	Octree of data.
in	<i>filename</i>	Input file name.
in	<i>filter</i>	Filtering algorithm of cells.

Returns

True on success, false on error.

6.31.2.13 template<class Position , class Extent , typename Integral , typename Real , class Parameter , class Conic , class Octree , class Function , class Element , class Index , class Data , unsigned int Dimension, class > bool Input::importhdf5 (const Parameter & parameters, unsigned int & rank, const std::array< std::array< double, 3 >, 3 > & rotm1, const double & thetay, const double & thetaz, const Conic & conic, Octree & octree, const std::string & filename, Function && filter) [static]

Ramsey importation.

Imports HDF5 data from ramsey gravity files. All cells selected by the provided filter are added to the octree.

Template Parameters

<i>Position</i>	Position.
<i>Extent</i>	Extent type.
<i>Integral</i>	Integral type of the file.
<i>Real</i>	Real type of the file.
<i>Conic</i>	Cone type.
<i>Octree</i>	Octree type.
<i>Function</i>	Function type taking an element as argument and returning a boolean.

<i>Element</i>	Underlying element type.
<i>Index</i>	Index type.
<i>Data</i>	Data type.
<i>Dimension</i>	Number of dimensions.

Parameters

in	<i>parameters</i>	Parameter structure.
in	<i>rank</i>	Rank.
in	<i>rotm1</i>	rotation matrix for narrow cones
in,out	<i>thetay</i>	Semi-angle for solid angle in direction y
in,out	<i>thetaz</i>	Semi-angle for solid angle in direction z
in	<i>conic</i>	Cone characteristics
in,out	<i>octree</i>	Octree of data.
in	<i>filename</i>	Input file name.
in	<i>filter</i>	Filtering algorithm of cells.

Returns

True on success, false on error.

6.31.2.14 template<class Octree , class > bool Input::load (Octree & octree, const std::string & filename) [static]

Load temporary cone file.

Loads a temporary cone file into an octree.

Template Parameters

<i>Octree</i>	Octree type.
---------------	--------------

Parameters

in,out	<i>octree</i>	Destination octree.
in	<i>filename</i>	File name.

Returns

True on success, false otherwise.

6.31.2.15 template<unsigned int Selection, class Octree , unsigned int Dimension, class Element , class Index , class Data , class Type , class > Type Input::mean (const Octree & octree, const Element & element, int level = -1) [inline], [static]

Mean value over cells.

Computes the average of the provided data in all surrounding cells which are normal.

Template Parameters

<i>Selection</i>	Index of the data to be computed.
<i>Octree</i>	Octree type.
<i>Dimension</i>	Number of dimensions.
<i>Element</i>	Underlying element type.
<i>Index</i>	Index type.

<i>Data</i>	Data type.
<i>Type</i>	Selected data type.

Parameters

in	<i>octree</i>	Input octree.
in	<i>element</i>	Input element.
in	<i>level</i>	Cell level for computation.

Returns

Averaged value.

```
6.31.2.16 template<unsigned int Selection, class Octree , unsigned int Dimension, class Element , class Index , class Data
, class Type , class > Data Input::meanAll ( const Octree & octree, const Element & element, int level = -1 )
[inline], [static]
```

Mean value over cells.

Computes the average of cell tuple information all surrounding cells which are normal.

Template Parameters

<i>Selection</i>	Index of the data to be computed.
<i>Octree</i>	Octree type.
<i>Dimension</i>	Number of dimensions.
<i>Element</i>	Underlying element type.
<i>Index</i>	Index type.
<i>Data</i>	Data type.
<i>Type</i>	Selected data type.

Parameters

in	<i>octree</i>	Input octree.
in	<i>element</i>	Input element.
in	<i>level</i>	Cell level for computation.

Returns

Averaged value of tuple.

```
6.31.2.17 template<class Container , class Element , class > Container Input::parse ( const std::string & filename, const
std::string & separator = "=", const std::string & comment = "#" ) [static]
```

Parameter file parsing.

Parses the provided parameter file and returns a map of parameters.

Template Parameters

<i>Container</i>	Output map type.
<i>Element</i>	Element type.

Parameters

in	<i>filename</i>	File name.
in	<i>separator</i>	Separator string.
in	<i>comment</i>	Comment string.

Returns

Map of parameters.

6.31.2.18 `std::pair< std::string, std::string > Input::partition (const std::string & text, const std::string & separator = "—") [inline], [static]`

Partition string.

Splits the string in two parts before and after the provided separator.

Parameters

in	<i>text</i>	Input text.
in	<i>separator</i>	Separator string.

Returns

Partitioned string.

6.31.2.19 `template<class List , class Octree , class Sphere , class Conic , unsigned int Dimension, class > bool Input::prepare (List & list, const Octree & octree, const Sphere & sphere, const Conic & conic) [static]`

File list preparation.

Adds to the list, the octree files which intersects the provided sphere and cone.

Template Parameters

<i>List</i>	File list type.
<i>Octree</i>	Octree type.
<i>Sphere</i>	Sphere type.
<i>Conic</i>	Cone type.
<i>Dimension</i>	Number of dimensions.

Parameters

in	<i>list</i>	File list.
in	<i>octree</i>	Input octree.
in	<i>sphere</i>	Geometrical sphere.
in	<i>cone</i>	Three dimensional cone.

Returns

True if some files have been added to the list, false otherwise.

6.31.2.20 `template<class Octree , class > bool Input::save (Octree & octree, const std::string & filename) [static]`

Save temporary cone file.

Saves the octree in a temporary cone file.

Template Parameters

<i>Octree</i>	Octree type.
---------------	--------------

Parameters

<i>in, out</i>	<i>octree</i>	Source octree.
<i>in</i>	<i>filename</i>	File name.

Returns

True on success, false otherwise.

6.31.2.21 template<class Extent , class Data , class Vector , typename Type , unsigned int Dimension, class > Data Input::schwarzschildify (const Data & *data*, const Vector & *center*, const Vector & *position*, const Type *mass*, const Type *length*) [inline], [static]

Data schwarzschild setup.

Converts a data to one of a schwarzschild configuration.

Template Parameters

<i>Extent</i>	Extent of the hyperoctree.
<i>Data</i>	Data type.
<i>Vector</i>	Vector type.
<i>Type</i>	Arithmetic type.
<i>Dimension</i>	Number of dimensions.

Parameters

<i>in</i>	<i>data</i>	Input data.
<i>in</i>	<i>center</i>	Cell center.
<i>in</i>	<i>position</i>	Mass position.
<i>in</i>	<i>mass</i>	Mass in SI units.
<i>in</i>	<i>length</i>	Spatial length in SI units.

Returns

Schwarzschild data.

6.31.2.22 template<class Octree , class Vector , typename Type , class Function , class... Dummy, class Element , unsigned int Dimension, class Extent , class > unsigned int Input::schwarzschildify (Octree & *octree*, const Vector & *position*, const Type *mass*, const Type *length*, Function && *refiner*, Dummy...) [inline], [static]

Octree schwarzschild setup.

Converts each data of the octree to one of a schwarzschild configuration.

Template Parameters

<i>Octree</i>	Octree type.
<i>Vector</i>	Vector type.
<i>Type</i>	Arithmetic type.

<i>Function</i>	Function type.
<i>Dummy</i>	Dummy type.
<i>Element</i>	Underlying element type.
<i>Dimension</i>	Number of dimensions.
<i>Extent</i>	Extent of the hyperoctree.

Parameters

in,out	<i>octree</i>	Octree of data.
in	<i>position</i>	Mass position in the octree.
in	<i>mass</i>	Mass in SI units.
in	<i>length</i>	Spatial length in SI units.
in	<i>refiner</i>	Refinement function taking a data and a level as arguments and returning true when a refinement should be triggered

Returns

Schwarzschild octree.

6.31.2.23 template<class Parameter , class Data , typename Type , class > Data Input::sistemize (const Parameter & parameters, const Data & data, const Type a, const Type h, const Type omegam, const Type lboxmpch) [inline], [static]

Data conversion to SI units.

Converts a data to one expressed in SI units.

Template Parameters

<i>Data</i>	Data type.
<i>Type</i>	Arithmetic type.

Parameters

in	<i>data</i>	Input data.
in	<i>a</i>	Scale factor.
in	<i>h</i>	Hubble parameter.
in	<i>omegam</i>	Density parameter of matter.
in	<i>lboxmpch</i>	Box size in megaparsecs times h.
in	<i>mpc</i>	Value of one megaparsec in SI units.
in	<i>rhoch2</i>	Value of critical density times h squared in SI units.

Returns

Data in SI units.

6.31.2.24 template<class Parameter , class Octree , typename Type , class Element , class Data , class > unsigned int Input::sistemize (const Parameter & parameters, Octree & octree, const Type h, const Type omegam, const Type lboxmpch) [inline], [static]

Octree conversion to SI units.

Converts each element of the octree to one based on data expressed in SI units.

Template Parameters

<i>Octree</i>	Octree type.
<i>Type</i>	Arithmetic type.
<i>Element</i>	Underlying element type.
<i>Data</i>	Data type.

Parameters

<i>in,out</i>	<i>octree</i>	Octree of data.
<i>in</i>	<i>h</i>	Hubble parameter.
<i>in</i>	<i>omegam</i>	Density parameter of matter.
<i>in</i>	<i>lboxmpch</i>	Box size in megaparsecs times h.
<i>in</i>	<i>mpc</i>	Value of one megaparsec in SI units.
<i>in</i>	<i>rhoch2</i>	Value of critical density times h squared in SI units.

Returns

Octree in SI units.

6.31.2.25 `std::string Input::trim (const std::string & text, const std::string & comment = "#") [inline], [static]`

Trim string.

Trims string from leading and trailing white spaces, from comments, and from inner multi white spaces.

Parameters

<i>in</i>	<i>text</i>	Input text.
<i>in</i>	<i>comment</i>	Comment string.

Returns

Trimmed string.

The documentation for this exception was generated from the following file:

- /data/home/mbreton/magrathea_pathfinder/src/[input.h](#)

6.32 Integrator Exception Reference

Integration utilities for raytracing.

```
#include <integrator.h>
```

Static Public Member Functions

Initialization

- template<typename Type = double, class Sphere , class Vector , typename Scalar , class Engine , class Distribution , unsigned int Dimension = Sphere::dimension(), class = typename std::enable_if<Dimension == 3>::type>
static [Photon](#)< Type, Dimension > [launch](#) (const Sphere &sphere, const [Cone](#)< Vector, Scalar > &cone, Engine &engine, Distribution &distribution)
Launch a photon in a cone.

- template<typename Type = double, class Sphere , class Container , class Vector , typename Scalar , class Engine , class Distribution , unsigned int Dimension = Sphere::dimension(), class = typename std::enable_if<(Dimension == 3) && (std::is_convertible<typename std::remove_cv<typename std::remove_reference<decltype(std::declval<Container>()[0])>::type>::type, Cone<Vector, Scalar> >::value)>::type>
static **Photon**< Type, Dimension > **launch** (const Sphere &sphere, const **Cone**< Vector, Scalar > &cone, const Container &cones, Engine &engine, Distribution &distribution)
Launch a photon in a serie of cones.
- template<typename Type , unsigned int Dimension = 3, class = typename std::enable_if<Dimension == 3>::type>
static **Photon**< Type, Dimension > **launch** (const Type xbegin, const Type ybegin, const Type zbegin, const Type xend, const Type yend, const Type zend)
Launch a photon going from a position to another.
- template<typename Type , unsigned int Dimension = 3, class = typename std::enable_if<Dimension == 3>::type>
static **Photon**< Type, Dimension > **launch** (const Type xbegin, const Type ybegin, const Type zbegin, const Type phi, const Type theta)
Launch a photon going from a position to a specified direction.
- template<bool Center = false, typename Type , unsigned int Dimension, class Container = std::vector<Photon<Type, Dimension> >, class = typename std::enable_if<Dimension == 3>::type>
static Container **launch** (const **Photon**< Type, Dimension > &photon, const unsigned int count, const Type angle, const Type rotation=Type())
Launch photons around one photon.

Computation

- template<int Order = ORDER, class Array , class Element , class Cosmology , class Octree , class Type , unsigned int Dimension = Octree::dimension(), class Data = typename std::tuple_element<1, decltype(Octree::element())>::type, class Position = decltype(-Octree::position()), class Extent = decltype(Octree::extent())>
static Array & **dphotondl** (Array &output, const Array &input, const Cosmology &cosmology, const Octree &octree, const Type length, std::vector< Element > &elemsTsc)

Derivative of a photon.

Evolution

- template<int Order = ORDER, bool RK4 = true, bool Verbose = false, class Cosmology , class Octree , class Type , class Trajectory , unsigned int Dimension = Octree::dimension(), class Element = typename std::remove_cv<typename std::remove_reference<decltype(std::declval<Trajectory>()[0])>::type>::type, class Data = typename std::tuple_element<1, decltype(Octree::element())>::type, class Core = decltype(Element::template type<1>()), unsigned int Size = std::tuple_size<Core>::value, class Position = decltype(Octree::position()), class Extent = decltype(Octree::extent()), class Point >
static Trajectory & **integrate** (Trajectory &trajectory, const Cosmology &cosmology, const Octree &octree, const Point &vobs, const Type length, const unsigned int nsteps=1)

Geodesics integration.

- template<int Order = ORDER, bool RK4 = true, bool Verbose = false, class Cosmology , class Octree , class Type , class Trajectory , unsigned int Dimension = Octree::dimension(), class Element = typename std::remove_cv<typename std::remove_reference<decltype(std::declval<Trajectory>()[0])>::type>::type, class Data = typename std::tuple_element<1, decltype(Octree::element())>::type, class Core = decltype(Element::template type<1>()), unsigned int Size = std::tuple_size<Core>::value, class Position = decltype(Octree::position()), class Extent = decltype(Octree::extent()), class Point = std::array<Type, 3>>
static Trajectory & **integrate** (Trajectory &trajectory, const std::string interpolation, const Type interpRef, const Cosmology &cosmology, const Octree &octree, const Point &vobs, const Type length, const unsigned int nsteps=1, const Point &kiTarget=Point())

Geodesics integration.

- template<int Order = ORDER, bool RK4 = true, bool Verbose = false, class Cosmology , class Octree , class Type , unsigned int Dimension, class Homogeneous = std::vector<Photon<Type, Dimension> >, class Point , class = typename std::enable_if<(-Dimension == 3) && (Dimension == Octree::dimension())>::type>
static **magrathea::Evolution**< Photon< Type, Dimension > > **propagate** (const **Photon**< Type, Dimension > &photon, const unsigned int count, const Type angle, const Type rotation, const std::string &interpolation, const Cosmology &cosmology, const Octree &octree, const Point &vobs, const Type length, const unsigned int nsteps=1, const Type amin=Type(), const std::string &filenames=std::string(), const Homogeneous &homogeneous=Homogeneous())

Propagation of a ray bundle.

Test

- static int [example \(\)](#)

6.32.1 Detailed Description

Integration utilities for raytracing.

Provides a list of integration routines for geodesics integration.

6.32.2 Member Function Documentation

6.32.2.1 template<int Order, class Array , class Element , class Cosmology , class Octree , class Type , unsigned int Dimension, class Data , class Position , class Extent > Array & Integrator::dphotondl (Array & *output*, const Array & *input*, const Cosmology & *cosmology*, const Octree & *octree*, const Type *length*, std::vector< Element > & *elemsTsc*) [static]

Derivative of a photon.

Computes the derivative of the core components of a photon.

Template Parameters

<i>Order</i>	Octree interpolation order : 0 for NGP, 1 for CIC, 2 for TSC or -1 for an homogeneous universe.
<i>Array</i>	Core array type.
<i>Element</i>	data element for neighbouring cells
<i>Cosmology</i>	Cosmology evolution type.
<i>Octree</i>	Octree type.
<i>Type</i>	Scalar type.
<i>Dimension</i>	Number of space dimension.
<i>Data</i>	Data type.
<i>Position</i>	Position of the hyperoctree center.
<i>Extent</i>	Extent of the hyperoctree.

Parameters

in,out	<i>output</i>	Output data.
in	<i>input</i>	Input data.
in	<i>cosmology</i>	Cosmology evolution.
in	<i>octree</i>	Octree.
in	<i>length</i>	Spatial length.
in	<i>elemsTsc</i>	Indexes of neighbouring cells

Returns

Reference to the output data.

6.32.2.2 static int Integrator::example() [static]

6.32.2.3 template<int Order, bool RK4, bool Verbose, class Cosmology , class Octree , class Type , class Trajectory , unsigned int Dimension, class Element , class Data , class Core , unsigned int Size, class Position , class Extent , class Point > Trajectory & Integrator::integrate (Trajectory & trajectory, const Cosmology & cosmology, const Octree & octree, const Point & vobs, const Type length, const unsigned int nsteps = 1) [static]

Geodesics integration.

Integrates the geodesics equation of a photon.

Template Parameters

<i>Order</i>	Octree interpolation order : 0 for NGP, 1 for CIC, 2 for TSC or -1 for an homogeneous universe.
<i>RK4</i>	Runge-kutta of fourth order or euler.
<i>Verbose</i>	Verbose mode for debug purposes.
<i>Cosmology</i>	Cosmology evolution type.
<i>Octree</i>	Octree type.
<i>Type</i>	Scalar type.
<i>Trajectory</i>	Trajectory type.
<i>Dimension</i>	Number of space dimension.
<i>Element</i>	Photon type.
<i>Data</i>	Data type.
<i>Core</i>	Core data type.
<i>Size</i>	Core size.
<i>Position</i>	Position of the hyperoctree center.
<i>Extent</i>	Extent of the hyperoctree.
<i>Point</i>	Point type.

Parameters

in,out	<i>trajectory</i>	Trajectory.
in	<i>cosmology</i>	Cosmology evolution.
in	<i>octree</i>	Octree.
in	<i>vobs</i>	Observer peculiar velocity, in SI
in	<i>length</i>	Spatial length in SI units.
in	<i>nsteps</i>	Number of lambda steps per grid.

Returns

Reference to the trajectory data.

6.32.2.4 template<int Order, bool RK4, bool Verbose, class Cosmology , class Octree , class Type , class Trajectory , unsigned int Dimension, class Element , class Data , class Core , unsigned int Size, class Position , class Extent , class Point > Trajectory & Integrator::integrate (Trajectory & trajectory, const std::string interpolation, const Type interpRef, const Cosmology & cosmology, const Octree & octree, const Point & vobs, const Type length, const unsigned int nsteps = 1, const Point & kiTarget = Point ()) [static]

Geodesics integration.

Integrates the geodesics equation of a photon until some condition.

Template Parameters

<i>Order</i>	Octree interpolation order : 0 for NGP, 1 for CIC, 2 for TSC or -1 for an homogeneous universe.
<i>RK4</i>	Runge-kutta of fourth order or euler.
<i>Verbose</i>	Verbose mode for debug purposes.
<i>Cosmology</i>	Cosmology evolution type.
<i>Octree</i>	Octree type.
<i>Type</i>	Scalar type.
<i>Trajectory</i>	Trajectory type.
<i>Dimension</i>	Number of space dimension.
<i>Element</i>	Photon type.
<i>Data</i>	Data type.
<i>Core</i>	Core data type.
<i>Size</i>	Core size.
<i>Position</i>	Position of the hyperoctree center.
<i>Extent</i>	Extent of the hyperoctree.
<i>Point</i>	Point type.

Parameters

<i>in,out</i>	<i>trajectory</i>	Trajectory.
<i>in</i>	<i>interpolation</i>	Which type of stop criterion for the integration
<i>in</i>	<i>interpRef</i>	Where to stop the integration
<i>in</i>	<i>cosmology</i>	Cosmology evolution.
<i>in</i>	<i>octree</i>	Octree.
<i>in</i>	<i>vobs</i>	Observer peculiar velocity, in SI
<i>in</i>	<i>length</i>	Spatial length in SI units.
<i>in</i>	<i>nsteps</i>	Number of lambda steps per grid.
<i>in</i>	<i>kiTarget</i>	Normal to the plane needed for a given photon.

Returns

Reference to the trajectory data.

```
6.32.2.5 template<typename Type , class Sphere , class Vector , typename Scalar , class Engine , class Distribution , unsigned int Dimension, class > Photon< Type, Dimension > Integrator::launch( const Sphere & sphere, const Cone< Vector, Scalar > & cone, Engine & engine, Distribution & distribution ) [static]
```

Launch a photon in a cone.

Launches a photon in the provided cone.

Template Parameters

<i>Type</i>	Photon type.
<i>Dimension</i>	Number of space dimension.
<i>Vector</i>	Position vector type.
<i>Scalar</i>	Scalar data type.
<i>Engine</i>	Random engine type.
<i>Distribution</i>	Random distribution type.

Parameters

<i>in</i>	<i>sphere</i>	Sphere to pick up the center and the surface.
<i>in</i>	<i>cone</i>	Current cone.
<i>in,out</i>	<i>engine</i>	Random engine.

in, out	<i>distribution</i>	Random distribution.
---------	---------------------	----------------------

Returns

Initial photon in the cone.

6.32.2.6 template<typename Type , class Sphere , class Container , class Vector , typename Scalar , class Engine , class Distribution , unsigned int Dimension, class > Photon< Type, Dimension > Integrator::launch (const Sphere & *sphere*, const Cone< Vector, Scalar > & *cone*, const Container & *cones*, Engine & *engine*, Distribution & *distribution*) [static]

Launch a photon in a serie of cones.

Launches a photon in the provided cone with the guarantee that this cone is the closest one comparatively to a list of cones.

Template Parameters

<i>Type</i>	Photon type.
<i>Dimension</i>	Number of space dimension.
<i>Sphere</i>	Sphere type.
<i>Container</i>	Container of cones type.
<i>Vector</i>	Position vector type.
<i>Scalar</i>	Scalar data type.
<i>Engine</i>	Random engine type.
<i>Distribution</i>	Random distribution type.

Parameters

in	<i>sphere</i>	Sphere to pick up the center and the surface.
in	<i>cone</i>	Current cone.
in	<i>cones</i>	List of cones to compare distance.
in, out	<i>engine</i>	Random engine.
in, out	<i>distribution</i>	Random distribution.

Returns

Initial photon in the cone.

6.32.2.7 template<typename Type , unsigned int Dimension, class > Photon< Type, Dimension > Integrator::launch (const Type *xbegin*, const Type *ybegin*, const Type *zbegin*, const Type *xend*, const Type *yend*, const Type *zend*) [static]

Launch a photon going from a position to another.

Launches a photon starting from a point and going to another one.

Template Parameters

<i>Type</i>	Photon type.
<i>Dimension</i>	Number of space dimension.

Parameters

in	<i>xbegin</i>	Starting x coordinate.
in	<i>ybegin</i>	Starting y coordinate.
in	<i>zbegin</i>	Starting z coordinate.
in	<i>xend</i>	Ending x coordinate.
in	<i>yend</i>	Ending y coordinate.
in	<i>zend</i>	Ending z coordinate.

Returns

Initial photon.

6.32.2.8 template<typename Type , unsigned int Dimension, class > Photon< Type, Dimension > Integrator::launch (const Type *xbegin*, const Type *ybegin*, const Type *zbegin*, const Type *phi*, const Type *theta*) [static]

Launch a photon going from a position to a specified direction.

Launches a photon starting from a point and going to specified direction.

Template Parameters

<i>Type</i>	Photon type.
<i>Dimension</i>	Number of space dimension.

Parameters

in	<i>xbegin</i>	Starting x coordinate.
in	<i>ybegin</i>	Starting y coordinate.
in	<i>zbegin</i>	Starting z coordinate.
in	<i>phi</i>	Phi angular coordinate.
in	<i>theta</i>	Theta angular coordinate.

Returns

Initial photon.

6.32.2.9 template<bool Center, typename Type , unsigned int Dimension, class Container , class > Container Integrator::launch (const Photon< Type, Dimension > & *photon*, const unsigned int *count*, const Type *angle*, const Type *rotation = Type ()*) [static]

Launch photons around one photon.

Launches a group of photon on a cone around one photon.

Template Parameters

<i>Center</i>	Adds the center photon in the resulting vector if true.
<i>Type</i>	Photon type.
<i>Dimension</i>	Number of space dimension.
<i>Container</i>	Container of photons type.

Parameters

in	<i>photon</i>	Central photon.
in	<i>count</i>	Number of photons to return.

in	<i>angle</i>	Half-angle at the cone vertex.
in	<i>rotation</i>	Arbitrary rotation to optionally apply on the resulting circle of photons.

Returns

Circle of photons.

6.32.2.10 template<int Order, bool RK4, bool Verbose, class Cosmology , class Octree , class Type , unsigned int Dimension, class Homogeneous , class Point , class > **magrathea::Evolution< Photon< Type, Dimension > >**
Integrator::propagate (const Photon< Type, Dimension > & photon, const unsigned int count, const Type angle,
const Type rotation, const std::string & interpolation, const Cosmology & cosmology, const Octree & octree, const
Point & vobs, const Type length, const unsigned int nsteps = 1, const Type amin = Type (), const std::string &
filenames = std::string (), const Homogeneous & homogeneous = Homogeneous ()) [static]

Propagation of a ray bundle.

Propagates a ray bundle calling the integrator for each photon.

Template Parameters

<i>Order</i>	Octree interpolation order : 0 for NGP, 1 for CIC, 2 for TSC or -1 for an homogeneous universe.
<i>RK4</i>	Runge-kutta of fourth order or euler.
<i>Verbose</i>	Verbose mode for debug purposes.
<i>Cosmology</i>	Cosmology evolution type.
<i>Octree</i>	Octree type.
<i>Type</i>	Scalar type.
<i>Dimension</i>	Number of space dimension.
<i>Homogeneous</i>	Homogeneous reference.
<i>Point</i>	point type.

Parameters

in	<i>photon</i>	Central photon initial data.
in	<i>count</i>	Number of other photons to use.
in	<i>angle</i>	Half-angle at the cone vertex.
in	<i>rotation</i>	Arbitrary rotation to optionally apply on the resulting circle of photons.
in	<i>interpolation</i>	Stop condition : redshift, a, t, r.
in	<i>cosmology</i>	Cosmology evolution.
in	<i>octree</i>	Octree.
in	<i>vobs</i>	Observer peculiar velocity, in SI
in	<i>length</i>	Spatial length in SI units.
in	<i>nsteps</i>	Number of lambda steps per grid.
in	<i>amin</i>	If different from zero, all photons should end by this value of a.
in	<i>filenames</i>	File names of the output. If empty, no output. If at least one percent sign, all trajectories are saved. Otherwise, only the central one is saved.
in	<i>Homogeneous</i>	Optional homogeneous trajectory. If not provided the angular diameter distance use the inhomogeneous value of a. If provided, the homogeneous value of a for the given radius is used.

Returns

Central photon trajectory.

The documentation for this exception was generated from the following file:

- /data/home/mbreton/magrathea_pathfinder/src/integrator.h

6.33 magrathea::LagrangianCategory Exception Reference

Category concept of lagrangian : data moving with the flow.

```
#include <lagrangiancategory.h>
```

Static Public Member Functions

Test

- static int [example \(\)](#)
Example function.

6.33.1 Detailed Description

Category concept of lagrangian : data moving with the flow.

Categorizes data as an lagrangian one. A lagrangian data is moving with the flow : this is a way of looking at fluid motion where the observer follows an individual fluid parcel as it moves through space and time.

6.33.2 Member Function Documentation

6.33.2.1 int magrathea::LagrangianCategory::example () [static]

Example function.

Tests and demonstrates the use of [LagrangianCategory](#).

Returns

0 if no error.

The documentation for this exception was generated from the following file:

- /data/home/mbreton/magrathea_pathfinder/src/magrathea/[lagrangiancategory.h](#)

6.34 Lensing Class Reference

Integration utilities for raytracing.

```
#include <lensing.h>
```

Static Public Member Functions

Initialization

- template<int Order = ORDER, bool RK4 = true, bool Verbose = false, class Parameter , class Point , class Cosmology , class Octree , class Type >

```
static std::array< std::array< double, 2 >, 2 > dbetadtheta (const Parameter &parameters, const Point &kiTargets, const Type interpRef, const Point &observer, const Type phi, const Type theta, const Type dist, const Cosmology &cosmology, const Octree &octree, const Point &vobs, const Type length)
```

Compute the jacobian for lensing.

- template<int Order = ORDER, bool RK4 = true, bool Verbose = false, class Parameter , class Point , class Cosmology , class Octree , class Type >
 static std::vector< std::array< std::array< double, 2 >, 2 > > **dbetadtheta** (const Parameter &**parameters**, const std::vector< Point > &**kiTargets**, const std::vector< Type > &**interpRef**, const Point &**observer**, const Type **phi**, const Type **theta**, const std::vector< Type > &**dist**, const Cosmology &**cosmology**, const Octree &**octree**, const Point &**vobs**, const Type **length**)

Compute the jacobian for lensing.

- template<int Order = ORDER, bool RK4 = true, bool Verbose = false, class Type , class Trajectory , template< typename Kind, class Index, class Data, unsigned int Dimension, class Position, class Extent, class Element, class Container > class Octree, typename Kind , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container >
 static std::array< std::array< double, 2 >, 2 > **dbetadtheta_infinitesimal** (const Type **distance**, const Trajectory &**trajectory**, const Octree< Kind, Index, Data, Dimension, Position, Extent, Element, Container > &**octree**, const Type **length**)

Compute the jacobian for lensing.

- template<int Order = ORDER, bool RK4 = true, bool Verbose = false, class Type , class Trajectory , template< typename Kind, class Index, class Data, unsigned int Dimension, class Position, class Extent, class Element, class Container > class Octree, typename Kind , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container >
 static std::vector< std::array< std::array< double, 2 >, 2 > > **dbetadtheta_infinitesimal** (const std::vector< Type > &**dist**, const Trajectory &**trajectory**, const Octree< Kind, Index, Data, Dimension, Position, Extent, Element, Container > &**octree**, const Type **length**)

Compute the jacobian for lensing.

Test

- static int **example** ()

6.34.1 Detailed Description

Integration utilities for raytracing.

Provides a list of integration routines for geodesics integration.

6.34.2 Member Function Documentation

- 6.34.2.1 template<int Order, bool RK4, bool Verbose, class Parameter , class Point , class Cosmology , class Octree , class Type > std::array< std::array< double, 2 >, 2 > **Lensing::dbetadtheta** (const Parameter & **parameters**, const Point & **kiTarget**, const Type **interpRef**, const Point & **observer**, const Type **philnit**, const Type **thetalnit**, const Type **dist**, const Cosmology & **cosmology**, const Octree & **octree**, const Point & **vobs**, const Type **length**) [static]

Compute the jacobian for lensing.

Compute the jacobian for lensing

Template Parameters

<i>Order</i>	Octree interpolation order : 0 for NGP, 1 for CIC, 2 for TSC or -1 for an homogeneous universe.
<i>RK4</i>	Runge-kutta of fourth order or euler.
<i>Verbose</i>	Verbose mode for debug purposes.
<i>Parameter</i>	Parameter type.
<i>Point</i>	point type.
<i>Cosmology</i>	cosmology type
<i>Octree</i>	octree type
<i>Type</i>	Scalar type.

Parameters

in	<i>parameters</i>	Parameter structure.
in	<i>kiTarget</i>	Vector normal to the plane needed to compute the distortion matrix.
in	<i>interpRef</i>	value for interpolation of the bundle.
in	<i>observer</i>	Point observer position
in	<i>philinit</i>	initial trajectory angle
in	<i>thetalninit</i>	initial trajectory angle
in	<i>dist</i>	distance at different evaluations of the matrix
in	<i>cosmology</i>	Cosmology evolution.
in	<i>octree</i>	Octree.
in	<i>vobs</i>	Observer peculiar velocity, in SI
in	<i>length</i>	Spatial length in SI units.

Returns

2x2 array A_{ij} giving the jacobian from seen angles to true angles

```
6.34.2.2 template<int Order, bool RK4, bool Verbose, class Parameter , class Point , class Cosmology , class Octree , class Type > std::vector< std::array< std::array< double, 2 >, 2 > > Lensing::dbetadtheta ( const Parameter & parameters, const std::vector< Point > & kiTargets, const std::vector< Type > & interpRefvec, const Point & observer, const Type philinit, const Type thetalninit, const std::vector< Type > & dist, const Cosmology & cosmology, const Octree & octree, const Point & vobs, const Type length ) [static]
```

Compute the jacobian for lensing.

Compute the jacobian for lensing

Template Parameters

<i>Order</i>	Octree interpolation order : 0 for NGP, 1 for CIC, 2 for TSC or -1 for an homogeneous universe.
<i>RK4</i>	Runge-kutta of fourth order or euler.
<i>Verbose</i>	Verbose mode for debug purposes.
<i>Parameter</i>	Parameter type.
<i>Point</i>	point type.
<i>Cosmology</i>	cosmology type
<i>Octree</i>	octree type
<i>Type</i>	Scalar type.

Parameters

in	<i>parameters</i>	Parameter structure.
in	<i>kiTargets</i>	Vector of vectors normal to the plane needed to compute the distortion matrix at some given surface.
in	<i>interpRefvec</i>	values for interpolation of the bundle.
in	<i>observer</i>	Point observer position
in	<i>philinit</i>	initial trajectory angle
in	<i>thetalninit</i>	initial trajectory angle
in	<i>dist</i>	distance at different evaluations of the matrix
in	<i>interpolation</i>	String of a given interpolation
in	<i>cosmology</i>	Cosmology evolution.
in	<i>octree</i>	Octree.
in	<i>vobs</i>	Observer peculiar velocity, in SI
in	<i>length</i>	Spatial length in SI units.
in	<i>nsteps</i>	Number of lambda steps per grid.

Returns

2x2 array Aij giving the jacobian from seen angles to true angles

```
6.34.2.3 template<int Order, bool RK4, bool Verbose, class Type , class Trajectory , template< typename Kind, class Index,
class Data, unsigned int Dimension, class Position, class Extent, class Element, class Container > class Octree,
typename Kind , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class
Container > std::array< std::array< double, 2 >, 2 > Lensing::dbetadtheta_infinitesimal ( const Type distance,
const Trajectory & trajectory, const Octree< Kind, Index, Data, Dimension, Position, Extent, Element, Container > &
octree, const Type length ) [static]
```

Compute the jacobian for lensing.

Compute the jacobian for lensing

Template Parameters

<i>Order</i>	Octree interpolation order : 0 for NGP, 1 for CIC, 2 for TSC or -1 for an homogeneous universe.
<i>RK4</i>	Runge-kutta of fourth order or euler.
<i>Verbose</i>	Verbose mode for debug purposes.
<i>Type</i>	Scalar type.
<i>Trajectory</i>	trajectory type.
<i>Octree</i>	octree type
<i>Type</i>	type type
<i>Kind</i>	Kind type
<i>Index</i>	index type
<i>Data</i>	data type
<i>Dimension</i>	Number of dimensions
<i>Position</i>	position type
<i>Extent</i>	extent type
<i>Element</i>	element type
<i>Container</i>	container type

Parameters

in	<i>distance</i>	scalar comoving distance to stop lensing integration
in	<i>phi</i>	initial trajectory angle
in	<i>theta</i>	initial trajectory angle
in	<i>trajectory</i>	Photon trajectory used to compute the jacobian matrix.
in	<i>octree</i>	Octree.
in	<i>length</i>	Spatial length in SI units.

Returns

2x2 array Aij giving the jacobian from seen angles to true angles

```
6.34.2.4 template<int Order, bool RK4, bool Verbose, class Type , class Trajectory , template< typename Kind, class Index,
class Data, unsigned int Dimension, class Position, class Extent, class Element, class Container > class Octree,
typename Kind , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class
Container > std::vector< std::array< std::array< double, 2 >, 2 > > Lensing::dbetadtheta_infinitesimal ( const
std::vector< Type > & dist, const Trajectory & trajectory, const Octree< Kind, Index, Data, Dimension, Position,
Extent, Element, Container > & octree, const Type length ) [static]
```

Compute the jacobian for lensing.

Compute the jacobian for lensing

Template Parameters

<i>Order</i>	Octree interpolation order : 0 for NGP, 1 for CIC, 2 for TSC or -1 for an homogeneous universe.
<i>RK4</i>	Runge-kutta of fourth order or euler.
<i>Verbose</i>	Verbose mode for debug purposes.
<i>Type</i>	Scalar type.
<i>Trajectory</i>	trajectory type.
<i>Octree</i>	octree type
<i>Kind</i>	Kind type
<i>Type</i>	type type
<i>Index</i>	index type
<i>Data</i>	data type
<i>Dimension</i>	Number of dimensions
<i>Position</i>	position type
<i>Extent</i>	extent type
<i>Element</i>	element type
<i>Container</i>	container type

Parameters

in	<i>dist</i>	vector of comoving distances to stop lensing integrations
in	<i>phi</i>	initial trajectory angle
in	<i>theta</i>	initial trajectory angle
in	<i>trajectory</i>	Photon trajectory used to compute the jacobian matrix.
in	<i>octree</i>	Octree.
in	<i>length</i>	Spatial length in SI units.

Returns

2x2 array Aij giving the jacobian from seen angles to true angles

6.34.2.5 static int Lensing::example() [static]

The documentation for this class was generated from the following file:

- /data/home/mbreton/magrathea_pathfinder/src/lensing.h

6.35 Miscellaneous Class Reference

```
#include <miscellaneous.h>
```

Static Public Member Functions

- static void [getFilesInDir](#) (const std::string dirName, std::vector< std::string > &fileNames)
Get list of files.
- static void [Tokenize](#) (const std::string &str, std::vector< std::string > &tokens, const std::string &delimiters="")
Tokenize string.
- template<class Vectored >
 static void [clear_shrink](#) (Vectored &vector)
Clear and shrink vector.

- template<typename Type >
 static void **fullclear_vector** (std::vector< Type > &vector)
Erase vector.
- template<class Function , typename Integer >
 static void **TicketizeFunction** (const Integer rank, const Integer ntasks, Function &&function)
IO ticket funtion.
- template<class Cone , template< unsigned int, class, typename > class Sphere, unsigned int Dimension, class Vector , typename Scalar , typename Integer >
 static void **GenerateFullskyCones** (const Integer ncones, std::vector< Cone > &cone, std::vector< Cone > &conelRot, Sphere< Dimension, Vector, Scalar > &sphere)
Generate fullsky cones.
- template<class Parameter , class Cone , template< unsigned int, class, typename > class Sphere, unsigned int Dimension, class Vector , typename Scalar >
 static void **GenerateNarrowCones** (const Parameter ¶meters, std::vector< Cone > &cone, std::vector< Cone > &conelRot, Sphere< Dimension, Vector, Scalar > &sphere, std::array< std::array< double, 3 >, 3 > &rotm1, Scalar &thetay, Scalar &thetaz)
Generate narrow cones.
- template<class Octree , class Filelist , typename Integer >
 static void **loadOctree** (const Integer icone, Octree &octree, Filelist &conefile)
Load octree.
- template<class Octree , class Cosmology , class Parameters , typename Real >
 static void **correctOctree** (Octree &octree, const Cosmology &cosmology, Parameters ¶meters, const Real h, const Real omegam, const Real lboxmpch, Real &amin)
Correct octree.
- template<class Vector , typename Scalar , class Container >
 static std::vector< std::array < double, 8 > > **getTargets** (const std::vector< std::array< double, 8 > > &posTargets, const Cone< Vector, Scalar > &cone, const Container &cones)
Get target.
- template<class Parameter , class Cone , typename Type1 >
 static void **fill_particles_vectors** (const Parameter ¶meters, const Cone &cone, const std::vector< std::string > &shellList, std::vector< Type1 > &pos_part, std::vector< Type1 > &force_part, std::vector< Type1 > &potential_part, std::vector< Type1 > &a_part, const double thetay, const double thetaz)
Rewrite particle in vector.
- template<template< typename Type, class Index, class Data, unsigned int Dimension, class Position, class Extent, class Element, class Container > class Octree, typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container >
 static void **VizualizeOctree** (const Octree< Type, Index, Data, Dimension, Position, Extent, Element, Container > &octree, const Type radius)
Write position of cells.
- template<typename Integer , class Parameters >
 static void **ReadFromCat** (const Integer icone, const Parameters ¶meters, std::vector< std::array< double, 18 > > &catalogue)
Read magrathea source catalogue.

6.35.1 Member Function Documentation

6.35.1.1 template<class Vectored > void Miscellaneous::clear_shrink (Vectored & vector) [static]

Clear and shrink vector.

Clear and shrink vector.

Template Parameters

<i>Vectored</i>	vector type
-----------------	-------------

Parameters

<code>in, out</code>	<code>vector</code>	Vector to be cleared.
----------------------	---------------------	-----------------------

Returns

Empty vector

6.35.1.2 template<class Octree , class Cosmology , class Parameters , typename Real > void Miscellaneous::correctOctree (Octree & octree, const Cosmology & cosmology, Parameters & parameters, const Real h, const Real omegam, const Real lboxmpch, Real & amin) [static]

Correct octree.

Process a loaded Octree.

Template Parameters

<code>Octree</code>	octree type
<code>Cosmology</code>	cosmology type
<code>Parameters</code>	parameters type
<code>Real</code>	float/real type

Parameters

<code>in, out</code>	<code>octree</code>	Octree to be filled
<code>in</code>	<code>cosmology</code>	Cosmological tables
<code>in</code>	<code>conefile</code>	Cone names in conedir
<code>in</code>	<code>parameters</code>	Parameters structure
<code>in</code>	<code>h</code>	dimensionless Hubble parameter
<code>in</code>	<code>omegam</code>	Matter density fraction
<code>in</code>	<code>lboxmpch</code>	Size of simulation box
<code>in, out</code>	<code>amin</code>	minimum value scale factor

6.35.1.3 template<class Parameter , class Cone , typename Type1 > void Miscellaneous::fill_particles_vectors (const Parameter & parameters, const Cone & cone, const std::vector< std::string > & shellList, std::vector< Type1 > & pos_part, std::vector< Type1 > & force_part, std::vector< Type1 > & potential_part, std::vector< Type1 > & a_part, const double thetay, const double thetaz) [static]

Rewrite particle in vector.

Rewrite particle position/velocity from HDF5 in a vector.

Template Parameters

<code>Parameter</code>	Parameter type
<code>Cone</code>	cone type
<code>Type1</code>	type1 type

Parameters

<code>in</code>	<code>parameters</code>	Parameter structure
<code>in</code>	<code>cone</code>	Cone parameters
<code>in</code>	<code>shellList</code>	List of HDF5 files containing particles
<code>in, out</code>	<code>pos_part</code>	Position of particles
<code>in, out</code>	<code>force_part</code>	Force of particles
<code>in, out</code>	<code>potential_part</code>	Potential of particles

<i>in,out</i>	<i>a_part</i>	scale factor of particles
<i>in</i>	<i>thetay</i>	Semi-angle for solid angle in direction y
<i>in</i>	<i>thetaz</i>	Semi-angle for solid angle in direction z

6.35.1.4 template<typename Type > void Miscellaneous::fullclear_vector (std::vector< Type > & vector) [static]

Erase vector.

Erase vector.

Template Parameters

Type	Type type
------	-----------

Parameters

<i>in,out</i>	<i>vector</i>	Vector to be erased.
---------------	---------------	----------------------

Returns

Empty vector

6.35.1.5 template<class Cone , template< unsigned int, class, typename > class Sphere, unsigned int Dimension, class Vector , typename Scalar , typename Integer > void Miscellaneous::GenerateFullskyCones (const Integer *ncones*, std::vector< Cone > & cone, std::vector< Cone > & coneffRot, Sphere< Dimension, Vector, Scalar > & sphere) [static]

Generate fullsky cones.

Generate fullsky cones.

Template Parameters

<i>Cone</i>	cone type
<i>Sphere</i>	sphere type
<i>Dimension</i>	dimension
<i>Vector</i>	vector type
<i>Scalar</i>	scalar type
<i>Integer</i>	integer type

Parameters

<i>in</i>	<i>ncones</i>	Number of cones .
<i>in,out</i>	<i>cone</i>	Cones generated
<i>in</i>	<i>coneffRot</i>	Cones generated
<i>in</i>	<i>sphere</i>	Central sphere

Returns

Generate the shape of fullsky cones.

```
6.35.1.6 template<class Parameter , class Cone , template< unsigned int, class, typename > class Sphere, unsigned int Dimension, class Vector , typename Scalar > void Miscellaneous::GenerateNarrowCones ( const Parameter & parameters, std::vector< Cone > & cone, std::vector< Cone > & conelRot, Sphere< Dimension, Vector, Scalar > & sphere, std::array< std::array< double, 3 >, 3 > & rotm1, Scalar & thetay, Scalar & thetaz ) [static]
```

Generate narrow cones.

Generate narrow cones.

Template Parameters

<i>Parameter</i>	Parameter type.
<i>Cone</i>	cone type
<i>Sphere</i>	sphere type
<i>Dimension</i>	dimension
<i>Vector</i>	vector type
<i>Scalar</i>	scalar type

Parameters

in	<i>parameters</i>	Parameter structure
in,out	<i>cone</i>	Cones generated
in,out	<i>conelRot</i>	Cones generated
in	<i>sphere</i>	Central sphere
in,out	<i>rotm1</i>	Rotation matrix for narrow cone cells
in,out	<i>thetay</i>	Semi-angle for solid angle in direction y
in,out	<i>thetaz</i>	Semi-angle for solid angle in direction z

```
6.35.1.7 void Miscellaneous::getFilesInDir ( const std::string dirName, std::vector< std::string > & fileNames ) [static]
```

Get list of files.

Get list of files and directories.

Parameters

in	<i>String</i>	Directory name.
in,out	<i>vector<string></i>	List of files and directories.

```
6.35.1.8 template<class Vector , typename Scalar , class Container > std::vector< std::array< double, 8 > > > Miscellaneous::getTargets ( const std::vector< std::array< double, 8 > > & posTargets, const Cone< Vector, Scalar > & cone, const Container & cones ) [static]
```

Get target.

Get target if inside cone.

Template Parameters

<i>Vector</i>	Position vector type.
<i>Scalar</i>	Scalar data type
<i>Container</i>	Container of cone type

Parameters

in	<i>posTargets</i>	vector of halo positions.
in	<i>Cone</i>	current cone.
in	<i>Cones</i>	to compare distances. return Vector vector of targets inside cone

6.35.1.9 template<class Octree , class Filelist , typename Integer > void Miscellaneous::loadOctree (const Integer *icone*, Octree & *octree*, Filelist & *conefile*) [static]

Load octree.

Load binary octree from preparation phase.

Template Parameters

<i>Octree</i>	octree type
<i>Filelist</i>	cone file type
<i>Integer</i>	integer type

Parameters

in	<i>icone</i>	cone number
in,out	<i>octree</i>	Octree to be filled
in	<i>conefile</i>	<i>Cone</i> names in conedir

6.35.1.10 template<typename Integer , class Parameters > void Miscellaneous::ReadFromCat (const Integer *icone*, const Parameters & *parameters*, std::vector< std::array< double, 18 > > & *catalogue*) [static]

Read magrathea source catalogue.

Read magrathea source catalogue.

Template Parameters

<i>Integer</i>	Integer type
<i>Parameter</i>	Parameter type

Parameters

in	<i>parameters</i>	Parameter structure
in	<i>icone</i>	cone number
in,out	<i>Catalogue</i>	Vector containing the source catalogue

6.35.1.11 template<class Function , typename Integer > void Miscellaneous::TicketizeFunction (const Integer *rank*, const Integer *ntasks*, Function && *function*) [static]

IO ticket funtion.

IO ticket system for heavy IO computations.

Template Parameters

<i>Function</i>	lambda function
<i>Integer</i>	integer type

Parameters

in	<i>rank</i>	Process rank .
in	<i>ntasks</i>	Number of tasks
in	<i>function</i>	Lambda function

Returns

Operation using ticketing system.

```
6.35.1.12 void Miscellaneous::Tokenize( const std::string & str, std::vector< std::string > & tokens, const std::string & delimiters = " " ) [static]
```

Tokenize string.

Tokenize strings with delimiters.

Template Parameters

<i>str</i>	String to tokenize
------------	--------------------

Parameters

in,out	<i>tokens</i>	Vector of strings.
in	<i>delimiters</i>	Delimiters used to tokenize String.

Returns

tokenized vector

```
6.35.1.13 template<template< typename Type, class Index, class Data, unsigned int Dimension, class Position, class Extent, class Element, class Container > class Octree, typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > void Miscellaneous::VizualizeOctree( const Octree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & octree, const Type radius ) [static]
```

Write position of cells.

Write position of cells under a given radius.

Template Parameters

<i>Octree</i>	octree type
<i>Type</i>	type type
<i>Index</i>	index type
<i>Data</i>	data type
<i>Dimension</i>	Number of dimensions
<i>Position</i>	position type
<i>Extent</i>	extent type
<i>Element</i>	element type
<i>Container</i>	container type

Parameters

in	<i>octree</i>	Octree to be filled
in	<i>radius</i>	Maximum radius at which we write cells (in Ramses Units)

The documentation for this class was generated from the following file:

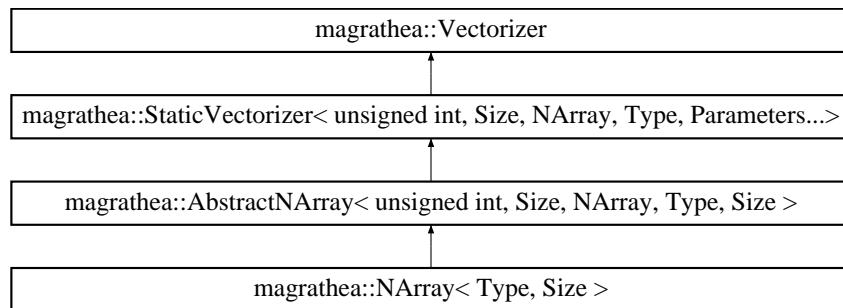
- /data/home/mbreton/magrathea_pathfinder/src/[miscellaneous.h](#)

6.36 magrathea::NArray< Type, Size > Exception Template Reference

Basic n-dimensional mathematical array.

```
#include <narray.h>
```

Inheritance diagram for magrathea::NArray< Type, Size >:



Public Member Functions

Lifecycle

- [NArray \(\)](#)
Implicit empty constructor.
- template<typename FundamentalType = Type, class = typename std::enable_if<std::is_fundamental<FundamentalType>::value>::type>
[NArray \(const NArray< FundamentalType, Size > &source\)](#)
Implicit conversion constructor.
- template<typename OtherType = Type, class... Misc, class = typename std::enable_if<std::is_convertible<OtherType, Type>::value>::type>
[NArray \(const std::initializer_list< OtherType > &source, const Misc &...misc\)](#)
Implicit initializer list constructor.
- template<class... Misc, class = typename std::enable_if<sizeof...(Misc) != 0>::type>
[NArray \(const Misc &...misc\)](#)
Explicit generic constructor.

Static Public Member Functions

Test

- static int [example \(\)](#)
Example function.

Public Attributes

- using [operator](#) = [typedef](#)

Protected Attributes

Data members

- Type `_data` [Size]
Data contents.

Additional Inherited Members

6.36.1 Detailed Description

`template<typename Type = double, unsigned int Size = 1>exception magrathea::NArray< Type, Size >`

Basic n-dimensional mathematical array.

This class is the direct derivation of [AbstractNArray](#). It provides the most basic n-dimensional mathematical array without adding new functionalities to the abstract class.

Template Parameters

<code>Type</code>	Data type.
<code>Size</code>	Number of elements.

6.36.2 Constructor & Destructor Documentation

6.36.2.1 `template<typename Type , unsigned int Size> magrathea::NArray< Type, Size >::NArray()` [inline]

Implicit empty constructor.

Does nothing.

6.36.2.2 `template<typename Type , unsigned int Size> template<typename FundamentalType , class > magrathea::NArray< Type, Size >::NArray(const NArray< FundamentalType, Size > & source)` [inline]

Implicit conversion constructor.

Provides an implicit conversion from a fundamental type contents.

Template Parameters

<code>FundamentalType</code>	(Fundamental data type.)
------------------------------	--------------------------

Parameters

<code>in</code>	<code>source</code>	Source of the copy.
-----------------	---------------------	---------------------

6.36.2.3 `template<typename Type , unsigned int Size> template<typename OtherType , class... Misc, class > magrathea::NArray< Type, Size >::NArray(const std::initializer_list< OtherType > & source, const Misc &... misc)` [inline]

Implicit initializer list constructor.

Provides an implicit conversion from an initializer list.

Template Parameters

<i>OtherType</i>	(Other data type.)
<i>Misc</i>	(Miscellaneous types.)

Parameters

in	<i>source</i>	Source of the copy.
in	<i>misc</i>	Miscellaneous arguments.

6.36.2.4 template<typename Type , unsigned int Size> template<class... Misc, class > **magrathea::NArray< Type, Size >::NArray** (const Misc &... *misc*) [inline], [explicit]

Explicit generic constructor.

Provides a generic interface to all constructors of the base class. Before calling the associated constructor of the base class, the contents is initialized.

Template Parameters

<i>Misc</i>	(Miscellaneous types.)
-------------	------------------------

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

6.36.3 Member Function Documentation

6.36.3.1 template<typename Type , unsigned int Size> int **magrathea::NArray< Type, Size >::example** () [static]

Example function.

Tests and demonstrates the use of **NArray**.

Returns

0 if no error.

6.36.4 Member Data Documentation

6.36.4.1 template<typename Type = double, unsigned int Size = 1> Type **magrathea::NArray< Type, Size >::_data[Size]** [protected]

Data contents.

6.36.4.2 template<typename Type = double, unsigned int Size = 1> using **magrathea::NArray< Type, Size >::operator =**

The documentation for this exception was generated from the following file:

- /data/home/mbreton/magrathea_pathfinder/src/magrathea/narray.h

6.37 Observer_velocity Class Reference

```
#include <observer_velocity.h>
```

Static Public Member Functions

- template<class Parameters , class Map >
static void ReadParamFile (Parameters ¶meters, Map ¶meter)
Read parameter file.
- template<typename Type1 , template< typename Type, class Index, class Data, unsigned int Dimension, class Position, class Extent, class Element, class Container > class Octree, typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container >
static void CreateOctreeVelocityWithCIC (Octree< Type, Index, Data, Dimension, Position, Extent, Element, Container > &octree, std::vector< Type1 > &pos_part, std::vector< Type1 > &vel_part)
Add velocity to Octree using CIC.
- template<typename Type1 , template< typename Type, class Index, class Data, unsigned int Dimension, class Position, class Extent, class Element, class Container > class Octree, typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container >
static void CreateOctreeVelocityWithTSC (Octree< Type, Index, Data, Dimension, Position, Extent, Element, Container > &octree, std::vector< Type1 > &pos_part, std::vector< Type1 > &vel_part)
Add velocity to Octree using TSC.

6.37.1 Member Function Documentation

6.37.1.1 template<typename Type1 , template< typename Type, class Index, class Data, unsigned int Dimension, class Position, class Extent, class Element, class Container > class Octree, typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > void
Observer_velocity::CreateOctreeVelocityWithCIC (Octree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & octree, std::vector< Type1 > & pos_part, std::vector< Type1 > & vel_part) [static]

Add velocity to Octree using CIC.

Add velocity to Octree using CIC

Template Parameters

<i>Type1</i>	Type1 type
<i>Octree</i>	octree type
<i>Type</i>	type type
<i>Index</i>	index type
<i>Data</i>	data type
<i>Dimension</i>	Number of dimensions
<i>Position</i>	position type
<i>Extent</i>	extent type
<i>Element</i>	element type
<i>Container</i>	container type

Parameters

<i>in,out</i>	<i>octree</i>	Octree be to filled with velocity field
<i>in</i>	<i>pos_part</i>	Position of particles
<i>in</i>	<i>vel_part</i>	Velocity of particles

6.37.1.2 template<typename Type1 , template< typename Type, class Index, class Data, unsigned int Dimension, class Position, class Extent, class Element, class Container > class Octree, typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > void Observer_velocity::CreateOctreeVelocityWithTSC (Octree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & octree, std::vector< Type1 > & pos_part, std::vector< Type1 > & vel_part) [static]

Add velocity to Octree using TSC.

Add velocity to Octree using TSC

Template Parameters

<i>Type1</i>	Type1 type
<i>Octree</i>	octree type
<i>Type</i>	type type
<i>Index</i>	index type
<i>Data</i>	data type
<i>Dimension</i>	Number of dimensions
<i>Position</i>	position type
<i>Extent</i>	extent type
<i>Element</i>	element type
<i>Container</i>	container type

Parameters

in, out	<i>octree</i>	Octree be to filled with velocity field
in	<i>pos_part</i>	Position of particles
in	<i>vel_part</i>	Velocity of particles

6.37.1.3 template<class Parameters , class Map > void Observer_velocity::ReadParamFile (Parameters & parameters, Map & parameter) [static]

Read parameter file.

Read and put in a structure the parameters.

Template Parameters

<i>Parameters</i>	structure type
<i>Map</i>	map type

Parameters

in, out	<i>parameters</i>	Structure containing the parameters.
in	<i>parameter</i>	Contains parameters to be rewritten

The documentation for this class was generated from the following file:

- /data/home/mbreton/magrathea_pathfinder/src/observer_velocity.h

6.38 Output Exception Reference

[Output](#) utilities for raytracing.

```
#include <output.h>
```

Static Public Member Functions

Utilities

- template<class Type = std::string, class = typename std::enable_if<std::is_convertible<typename std::remove_cv<typename std::remove_reference<Type>::type, std::string>::value>::type>
 static std::string **name** (Type &&value=Type())

File name from a string.
- template<class Type , class = typename std::enable_if<!std::is_convertible<typename std::remove_cv<typename std::remove_reference<Type>::type, std::string>::value>::type>
 static std::string **name** (const Type &value)

File name from a number.
- template<template< class, class > class Type , class First , class Second , class = typename std::enable_if<(std::tuple_size<Type<First, Second> >::value == std::tuple_size<std::pair<First, Second> >::value) && (std::is_convertible<First, std::string>-::value)>::type>
 static std::string **name** (const Type< First, Second > &value)

File name from a format.
- template<class Type , class... Types, class = typename std::enable_if<sizeof...(Types) != 0>::type>
 static std::string **name** (Type &&value, Types &&...values)

File name from a format.

Save

- template<class Octree , class = typename std::enable_if<Octree::dimension() != 0>::type>
 static bool **save** (std::ostream &stream, const Octree &octree, const int digits=0)

Save a octree.
- template<class Cosmology , class = typename std::enable_if<std::tuple_size<Cosmology>::value != 0>::type>
 static bool **save** (std::ostream &stream, const Cosmology &cosmology, const unsigned int digits=0)

Save a cosmology.
- template<class Trajectory , class = typename std::enable_if<!std::is_void<typename std::remove_cv<typename std::remove_reference<decltype(std::declval<Trajectory>()[0].type())>::type>::value>::type>
 static bool **save** (std::ostream &stream, const Trajectory &trajectory, const unsigned int &digits=0)

Save a trajectory.
- template<class Container , typename Type = typename std::remove_cv<typename std::remove_reference<decltype(std::declval<Container>()[0])>::type>, typename Integral = std::true_type, class = typename std::enable_if<!std::is_void<typename std::remove_cv<typename std::remove_reference<decltype(std::declval<Container>()[0])>::type>::value>::type>
 static bool **save** (std::ostream &stream, const Container &x, const Container &y, const Container &ymean, const Container &ystd, const unsigned int digits=0, const Integral count=Integral())

Save statistics.

Test

- static int **example** ()

Example function.

6.38.1 Detailed Description

[Output](#) utilities for raytracing.

Provides a list of exportation routines to save data for raytracing.

6.38.2 Member Function Documentation

6.38.2.1 int Output::example() [static]

Example function.

Tests and demonstrates the use of [Output](#).

Returns

0 if no error.

6.38.2.2 template<class Type , class > std::string Output::name (Type && value = Type ()) [inline], [static]

File name from a string.

Produces a file name from a string.

Template Parameters

Type	Type convertible to a string.
------	-------------------------------

Parameters

in	value	Value of the string.
----	-------	----------------------

Returns

Name corresponding to the string.

6.38.2.3 template<class Type , class > std::string Output::name (const Type & value) [inline], [static]

File name from a number.

Produces a file name from a number.

Template Parameters

Type	Number type convertible to a string.
------	--------------------------------------

Parameters

in	value	Value of the number.
----	-------	----------------------

Returns

Name corresponding to the number.

6.38.2.4 template<template< class, class > class Type, class First , class Second , class > std::string Output::name (const Type< First, Second > & value) [inline], [static]

File name from a format.

Produces a file name from a format.

Template Parameters

Type	Pair type.
First	Type convertible to a string.
Second	Second type associated to the format.

Parameters

in	value	Value of the pair.
----	-------	--------------------

Returns

Name corresponding to the format.

6.38.2.5 template<class Type, class... Types, class > std::string Output::name (Type && value, Types &&... values) [inline], [static]

File name from a format.

Produces a file name from a serie of components by recursively calling the name function.

Template Parameters

Type	First type.
Types	Other types.

Parameters

in	value	First value.
in	values	Other values.

Returns

Name corresponding to the serie of components.

6.38.2.6 template<class Octree , class > bool Output::save (std::ostream & stream, const Octree & octree, const int digits = 0) [static]

Save a octree.

Writes an octree to a text file.

Template Parameters

Octree	Octree type.
--------	--------------

Parameters

in,out	stream	Output stream.
in	octree	Octree.
in	digits	Optional precision.

Returns

True on success, false otherwise.

6.38.2.7 template<class Cosmology , class > bool Output::save (std::ostream & stream, const Cosmology & cosmology, const unsigned int digits = 0) [static]

Save a cosmology.

Writes each step of a cosmology to a text file.

Template Parameters

<i>Cosmology</i>	Cosmology type.
------------------	-----------------

Parameters

<i>in,out</i>	<i>stream</i>	Output stream.
<i>in</i>	<i>cosmology</i>	Cosmology.
<i>in</i>	<i>digits</i>	Optional precision.

Returns

True on success, false otherwise.

6.38.2.8 template<class Trajectory , class > bool Output::save (std::ostream & *stream*, const Trajectory & *trajectory*, const unsigned int & *digits* = 0) [static]

Save a trajectory.

Writes each step of the trajectory to a text file.

Template Parameters

<i>Trajectory</i>	Trajectory type.
-------------------	------------------

Parameters

<i>in,out</i>	<i>stream</i>	Output stream.
<i>in</i>	<i>trajectory</i>	Trajectory.
<i>in</i>	<i>digits</i>	Optional precision.

Returns

True on success, false otherwise.

6.38.2.9 template<class Container , typename Type , typename Integral , class > bool Output::save (std::ostream & *stream*, const Container & *x*, const Container & *y*, const Container & *ymean*, const Container & *ystd*, const unsigned int *digits* = 0, const Integral *count* = Integral()) [static]

Save statistics.

Save photons statistics.

Template Parameters

<i>Container</i>	Container type.
<i>Type</i>	Data type.
<i>Integral</i>	Integral type.

Parameters

<i>in,out</i>	<i>stream</i>	Output stream.
<i>in</i>	<i>x</i>	Abscissae.
<i>in</i>	<i>y</i>	Homogeneous ordinates.
<i>in</i>	<i>ymean</i>	Mean of values.
<i>in</i>	<i>ystd</i>	Standard deviation of values.

<code>in</code>	<code>digits</code>	Optional precision.
<code>in</code>	<code>count</code>	Optional count.

Returns

True on success, false otherwise.

The documentation for this exception was generated from the following file:

- /data/home/mbreton/magrathea_pathfinder/src/output.h

6.39 parameters_t Struct Reference

```
#include <catalogues.h>
```

Public Attributes

- `std::string celldir`
- `std::string conedir`
- `std::string conefmt`
- `std::string evolfile`
- `uint isfullsky`
- `real mpc`
- `uint ncoarse`
- `uint nc ones`
- `std::string paramfile`
- `real rhoch2`
- `uint seed`
- `uint typefile`
- `std::string base`
- `real cat_accuracy`
- `uint firstcone`
- `uint halos`
- `std::string jacobiantype`
- `uint lastcone`
- `uint npart`
- `uint nsteps`
- `real openingmin`
- `std::string outputdir`
- `std::string outputprefix`
- `std::string plane`
- `std::string sourcedir`
- `std::string stop_bundle`
- `uint use_previous_catalogues`
- `real v0x`
- `real v0y`
- `real v0z`
- `real zmin`
- `real zmax`
- `uint acorrection`
- `uint allocation`
- `std::string cellfmt`

- `uint coarsecorrection`
- `uint coarseonly`
- `uint correction`
- `std::string inputtype`
- `uint microcoeff`
- `std::string minicone`
- `std::string partdir`
- `std::string map_components`
- `uint nb_z_maps`
- `uint nbundlemin`
- `long nside`
- `std::string stop_ray`
- `real z_stop_min`
- `real z_stop_max`
- `std::string velocity_field_v0`
- `uint makestat`
- `uint massmin`
- `uint massmax`
- `uint nbundlecnt`
- `uint nstat`
- `uint ntrajectories`
- `uint openingcnt`
- `std::string ray_targets`
- `integer savemode`
- `std::string statistic`

6.39.1 Member Data Documentation

6.39.1.1 `uint parameters_t::acorrection`

6.39.1.2 `uint parameters_t::allocation`

6.39.1.3 `std::string parameters_t::base`

6.39.1.4 `real parameters_t::cat_accuracy`

6.39.1.5 `std::string parameters_t::celldir`

6.39.1.6 `std::string parameters_t::cellfmt`

6.39.1.7 `uint parameters_t::coarsecorrection`

6.39.1.8 `uint parameters_t::coarseonly`

6.39.1.9 `std::string parameters_t::conedir`

6.39.1.10 `std::string parameters_t::conefmt`

6.39.1.11 `uint parameters_t::correction`

6.39.1.12 `std::string parameters_t::evolfile`

6.39.1.13 `uint parameters_t::firstcone`

- 6.39.1.14 `uint parameters_t::halos`
- 6.39.1.15 `std::string parameters_t::inputtype`
- 6.39.1.16 `uint parameters_t::isfullsky`
- 6.39.1.17 `std::string parameters_t::jacobiantype`
- 6.39.1.18 `uint parameters_t::lastcone`
- 6.39.1.19 `uint parameters_t::makestat`
- 6.39.1.20 `std::string parameters_t::map_components`
- 6.39.1.21 `uint parameters_t::massmax`
- 6.39.1.22 `uint parameters_t::massmin`
- 6.39.1.23 `uint parameters_t::microcoeff`
- 6.39.1.24 `std::string parameters_t::minicone`
- 6.39.1.25 `real parameters_t::mpc`
- 6.39.1.26 `uint parameters_t::nb_z_maps`
- 6.39.1.27 `uint parameters_t::nbundlecnt`
- 6.39.1.28 `uint parameters_t::nbundlemin`
- 6.39.1.29 `uint parameters_t::ncoarse`
- 6.39.1.30 `uint parameters_t::ncones`
- 6.39.1.31 `uint parameters_t::npart`
- 6.39.1.32 `long parameters_t::nside`
- 6.39.1.33 `uint parameters_t::nstat`
- 6.39.1.34 `uint parameters_t::nsteps`
- 6.39.1.35 `uint parameters_t::ntrajectories`
- 6.39.1.36 `uint parameters_t::openingcnt`
- 6.39.1.37 `real parameters_t::openingmin`
- 6.39.1.38 `std::string parameters_t::outputdir`
- 6.39.1.39 `std::string parameters_t::outputprefix`
- 6.39.1.40 `std::string parameters_t::paramfile`
- 6.39.1.41 `std::string parameters_t::partdir`

- 6.39.1.42 std::string parameters_t::plane
- 6.39.1.43 std::string parameters_t::ray_targets
- 6.39.1.44 real parameters_t::rhoch2
- 6.39.1.45 integer parameters_t::savemode
- 6.39.1.46 uint parameters_t::seed
- 6.39.1.47 std::string parameters_t::sourcedir
- 6.39.1.48 std::string parameters_t::statistic
- 6.39.1.49 std::string parameters_t::stop_bundle
- 6.39.1.50 std::string parameters_t::stop_ray
- 6.39.1.51 uint parameters_t::typefile
- 6.39.1.52 uint parameters_t::use_previous_catalogues
- 6.39.1.53 real parameters_t::v0x
- 6.39.1.54 real parameters_t::v0y
- 6.39.1.55 real parameters_t::v0z
- 6.39.1.56 std::string parameters_t::velocity_field_v0
- 6.39.1.57 real parameters_t::z_stop_max
- 6.39.1.58 real parameters_t::z_stop_min
- 6.39.1.59 real parameters_t::zmax
- 6.39.1.60 real parameters_t::zmin

The documentation for this struct was generated from the following files:

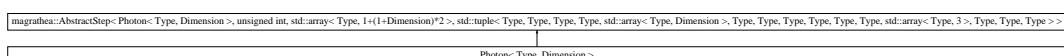
- /data/home/mbreton/magrathea_pathfinder/src/[catalogues.h](#)
- /data/home/mbreton/magrathea_pathfinder/src/[create_octree.h](#)
- /data/home/mbreton/magrathea_pathfinder/src/[hmaps.h](#)
- /data/home/mbreton/magrathea_pathfinder/src/[observer_velocity.h](#)
- /data/home/mbreton/magrathea_pathfinder/src/[rays.h](#)

6.40 Photon< Type, Dimension > Exception Template Reference

[Photon](#) step implementation for raytracing.

```
#include <photon.h>
```

Inheritance diagram for Photon< Type, Dimension >:



Public Member Functions

Lifecycle

- template<class... Misc>
Photon (Misc &&...misc)
Explicit generic constructor.

Data

- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<magrathea::AbstractStep<Photon<-Type, Dimension>, unsigned int, std::array<Type, 1+(1+Dimension)*2>, std::tuple<Type, Type, Type, Type, std::array<Type, Dimension>, Type, Type, Type, Type, Type, std::array<Type, 3>, Type, Type, Type> >>().template id<Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type>
Template index (Misc &&...misc)
Access to the index data.
- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<const magrathea::AbstractStep<Photon<-Type, Dimension>, unsigned int, std::array<Type, 1+(1+Dimension)*2>, std::tuple<Type, Type, Type, Type, std::array<Type, Dimension>, Type, Type, Type, Type, Type, std::array<Type, 3>, Type, Type, Type> >>().template id<Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type>
Template index (Misc &&...misc) const
Immutable access to the index data.
- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<magrathea::AbstractStep<Photon<-Type, Dimension>, unsigned int, std::array<Type, 1+(1+Dimension)*2>, std::tuple<Type, Type, Type, Type, std::array<Type, Dimension>, Type, Type, Type, Type, Type, std::array<Type, 3>, Type, Type, Type> >>().template core<0, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type>
Template a (Misc &&...misc)
Access to the a data.
- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<const magrathea::AbstractStep<Photon<-Type, Dimension>, unsigned int, std::array<Type, 1+(1+Dimension)*2>, std::tuple<Type, Type, Type, Type, std::array<Type, Dimension>, Type, Type, Type, Type, Type, std::array<Type, 3>, Type, Type, Type> >>().template core<0, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type>
Template a (Misc &&...misc) const
Immutable access to the a data.
- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<magrathea::AbstractStep<Photon<-Type, Dimension>, unsigned int, std::array<Type, 1+(1+Dimension)*2>, std::tuple<Type, Type, Type, Type, std::array<Type, Dimension>, Type, Type, Type, Type, Type, std::array<Type, 3>, Type, Type, Type> >>().template core<1, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type>
Template t (Misc &&...misc)
Access to the t data.
- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<const magrathea::AbstractStep<Photon<-Type, Dimension>, unsigned int, std::array<Type, 1+(1+Dimension)*2>, std::tuple<Type, Type, Type, Type, std::array<Type, Dimension>, Type, Type, Type, Type, Type, std::array<Type, 3>, Type, Type, Type> >>().template core<1, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type>
Template t (Misc &&...misc) const
Immutable access to the t data.
- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<magrathea::AbstractStep<Photon<-Type, Dimension>, unsigned int, std::array<Type, 1+(1+Dimension)*2>, std::tuple<Type, Type, Type, Type, std::array<Type, Dimension>, Type, Type, Type, Type, Type, std::array<Type, 3>, Type, Type, Type> >>().template core<2, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type>
Template x (Misc &&...misc)
Access to the x data.
- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<const magrathea::AbstractStep<Photon<-Type, Dimension>, unsigned int, std::array<Type, 1+(1+Dimension)*2>, std::tuple<Type, Type, Type, Type, std::array<Type, Dimension>, Type, Type, Type, Type, Type, std::array<Type, 3>, Type, Type, Type> >>().template core<2, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type>
Template x (Misc &&...misc) const
Immutable access to the x data.

Immutable access to the dyld data.

- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<magrathea::AbstractStep<Photon<-Type, Dimension>, unsigned int, std::array<Type, 1+(1+Dimension)*2>, std::tuple<Type, Type, Type, Type, std::array<Type, Dimension>, Type, Type, Type, Type, Type, std::array<Type, 3>, Type, Type, Type>>().template core<8, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type> Template [dzdl](#) (Misc &...misc)

Access to the dzdl data

- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<const magrathea::AbstractStep<Photon<-Type, Dimension>, unsigned int, std::array<Type, 1+(1+Dimension)*2>, std::tuple<Type, Type, Type, Type, std::array<Type, Dimension>, Type, Type, Type, Type, Type, std::array<Type, 3>, Type, Type, Type>>().template core<8, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<std::is_void<Template>::value>::type>

Template `azul` (`Misc &&...Misc`) const
Immutabile accesso alla `stack` data.

Immutable access to the `azul` data.

- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<magrathea::AbstractStep<Photon<-Type, Dimension>, unsigned int, std::array<Type, 1+(1+Dimension)*2>, std::tuple<Type, Type, Type, Type, std::array<Type, Dimension>, Type, Type, Type, Type, Type, std::array<Type, 3>, Type, Type, Type>>>().template extra<0, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type
Template level (Misc &&...misc)

Access to the level data.

template<unsigned int... Values>

- Type, Dimension>, unsigned int, std::array<Type, 1+(1+Dimension)*2>, std::tuple<Type, Type, Type, Type, Type, std::array<Type, Dimension>, Type, Type, Type, Type, Type, std::array<Type, 3>, Type, Type, Type>>>().template extra<0, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type>
Template level (Misc &&...misc) const

Immutable access to the level data.

template<unsigned int... Values, class...

- Type, Dimension>, unsigned int, std::array<Type, 1+(1+Dimension)*2>, std::tuple<Type, Type, Type, Type, Type, std::array<Type, Dimension>, Type, Type, Type, Type, Type, std::array<Type, 3>, Type, Type, Type>>>().template extra<1, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type
Template **ah** (Misc &&...misc)

Access to the ah data.

template<unsigned int... Value>

- template <extra1<Values...>>(<extra2<Misc>>(...), class = typename std::enable_if<!std::is_void<Template>::value>::type>
Template ah (Misc &&...misc) const

Template an (Alice auth message) construct

template<unsigned int Values class

- ```
template<unsigned int... Values, class... Misc, class Type> = decltype(std::declval<ImaginicalAbstractStep<Notion<Type, Dimension>, unsigned int, std::array<Type, 1+(1+Dimension)*2>, std::tuple<Type, Type, Type, Type, std::array<Type, Dimension>, Type, Type, Type, Type, Type, std::array<Type, 3>, Type, Type, Type>>().template extra<2, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type>
```

#### Template **M5** (Misc &...misc)

Access to the MO data

- ```
template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<const magrittra::AbstractStep<Photon><Type, Dimension>(), unsigned int, std::array<Type, 1+(1+Dimension)*2>, std::tuple<Type, Type, Type, Type, std::array<Type, Dimension>, Type, Type, Type, Type, Type, std::array<Type, 3>, Type, Type, Type> >>().template extra<2, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type>
```

Template (Misc &&...Misc) const
Immutability access to the the data

Immutable access to the rho data.

- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<magrathea::AbstractStep<Photon<-Type, Dimension>, unsigned int, std::array<Type, 1+(1+Dimension)*2>, std::tuple<Type, Type, Type, Type, std::array<Type, Dimension>, Type, Type, Type, Type, Type, Type, std::array<Type, 3>, Type, Type, Type>>().template extra<3, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type> Template_phi (Misc && misc)

Template phi (Misc &&...misc)

Access to the phi data.
http://tinyurl.com/Ma

- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<const magrathea::AbstractStep<Photon<<Type, Dimension>, unsigned int, std::array<Type, 1+(1+Dimension)*2>, std::tuple<Type, Type, Type, Type, std::array<Type, Dimension>, Type, Type, Type, Type, Type, Type, std::array<Type, 3>, Type, Type, Type>>>().template extra<3, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type>

Template **phi** (Misc &&...misc) const

Immutable access to the laplacian data.

- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<magrathea::AbstractStep<Photon<-Type, Dimension>, unsigned int, std::array<Type, 1+(1+Dimension)*2>, std::tuple<Type, Type, Type, Type, std::array<Type, Dimension>, Type, Type, Type, Type, Type, std::array<Type, 3>, Type, Type, Type>>().template extra<7, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<std::is_void<Template>::value>::type> Template **redshift** (Misc && misc)

Template Redshift (Misc &&...misc)

Access to the redshift data.

- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<const magrathea::AbstractStep<Photon<-Type, Dimension>, unsigned int, std::array<Type, 1+(1+Dimension)*2>, std::tuple<Type, Type, Type, Type, std::array<Type, Dimension>, Type, Type, Type, Type, Type, std::array<Type, 3>, Type, Type, Type>>>().template extra<7, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<std::is_void<Template>::value>::type>

Immutable access to the redshift data

template< unsigned int Values class M

- ```
template<unsigned int... Values, class... Misc, class Type> = decltype(std::declval<magrathea::AbstractStep<1>()>.
Type, Dimension>, unsigned int, std::array<Type, 1+(1+Dimension)*2>, std::tuple<Type, Type, Type, Type, std::array<Type,
Dimension>, Type, Type, Type, Type, Type, std::array<Type, 3>, Type, Type, Type>>().template extra<8, Values...>(std-
::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type
Template dsdl2 (Misc &&...misc)
```

## *Access to the dsdl2 data.*

template<unsigned int... Values

- ```
Type, Dimension>, unsigned int, std::array<Type, 1+(1+Dimension)*2>, std::tuple<Type, Type, Type, Type, Type, std::array<Type, Dimension>, Type, Type, Type, Type, Type, std::array<Type, 3>, Type, Type, Type>>>().template extra<8, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<std::is_void<Template>::value>::type>
Template dsdl2 (Misc &&...misc) const
```

Immutable access to the dsdl2 data.

template<unsigned int... Values, class...

- Type, Dimension>, unsigned int, std::array<Type, 1+(1+Dimension)*2>, std::tuple<Type, Type, Type, Type, Type, std::array<Type, Dimension>, Type, Type, Type, Type, Type, Type, std::array<Type, 3>, Type, Type, Type>>>().template extra<9, Values...>(<std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type>

Access to the error data

template<unsigned int... Values>

- ```
template <unsigned int... Values, class... Misc, class Template = decltype(std::declval<const magical::AbstractType>().
Type, Dimension>, unsigned int, std::array<Type, 1+(1+Dimension)*2>, std::tuple<Type, Type, Type, Type, std::array<Type,
Dimension>, Type, Type, Type, Type, Type, std::array<Type, 3>, Type, Type, Type>>().template extra<9, Values...>(std-
::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type
Template error (Misc &&...misc) const
```

*Immutable access to the error data.*

template<unsigned int... Values, class...

- Type, Dimension>, unsigned int, std::array<Type, 1+(1+Dimension)\*2>, std::tuple<Type, Type, Type, Type, Type, std::array<Type, Dimension>, Type, Type, Type, Type, Type, std::array<Type, 3>, Type, Type, Type>>>().template extra<10, Values...>(std::declval<Misc>(...)), class = typename std::enable\_if<!std::is\_void<Template>::value>::type>

Template **distance** (Misc &...misc)

### *Access to the distance data.*

template<unsigned int... Values, class

- ```
Type, Dimension>, unsigned int, std::array<Type, 1+(1+Dimension)*2>, std::tuple<Type, Type, Type, Type, Type, std::array<Type, Dimension>, Type, Type, Type, Type, Type, std::array<Type, 3>, Type, Type, Type>>>().template extra<10, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type>
```

Immutable access to the distance data

template<unsigned int... Values, class... Mi

- template <designated with ...values, class...Misc, class template ...> std::type_traits::is_movable<Implementation>::value <Type, Dimension>, unsigned int, std::array<Type, 1+(1+Dimension)*2>, std::tuple<Type, Type, Type, Type, std::array<Type, Dimension>, Type, Type, Type, Type, Type, std::array<Type, 3>, Type, Type, Type>>().template extra<11, Values...>(0, std::declval<Misc>(...)), class = typename std::enable_if<std::is_void<Template>::value>::type> Template [isw](#) (Misc &&...misc)

Access to the isw data.

template<unsigned int... Value>

- Type, Dimension>, unsigned int, std::array<Type, 1+(1+Dimension)*2>, std::tuple<Type, Type, Type, Type, std::array<Type, Dimension>, Type, Type, Type, Type, Type, std::array<Type, 3>, Type, Type, Type>>().template extra<11, Values...>(0, std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type>
Template [isw](#) (Misc &&... misc) const

Template **ISW** (Misc &&...misc) const

Immutable access to the isw data.

- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<magrathea::AbstractStep<Photon<-Type, Dimension>, unsigned int, std::array<Type, 1+(1+Dimension)*2>, std::tuple<Type, Type, Type, Type, std::array<Type, Dimension>, Type, Type, Type, Type, Type, std::array<Type, 3>, Type, Type, Type>>>.template extra<11, Values...>(1, std::declval<Misc>(...)), class = typename std::enable_if<std::is_void<Template>::value>::type> Template is void (Misc 8.8 - misc)

Template ISWOLD (MISC &&...Misc)

Access to the iswold data.

- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<const magrathea::AbstractStep<Photon<-Type, Dimension>, unsigned int, std::array<Type, 1+(1+Dimension)*2>, std::tuple<Type, Type, Type, Type, std::array<Type, Dimension>, Type, Type, Type, Type, Type, std::array<Type, 3>, Type, Type, Type>>().template extra<11, Values...>(1, std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type>

Template `iswold` (Misc &&...misc) const

Immutable access to the iswold data.

- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<magrathea::AbstractStep<Photon<-Type, Dimension>, unsigned int, std::array<Type, 1+(1+Dimension)*2>, std::tuple<Type, Type, Type, Type, std::array<Type, Dimension>, Type, Type, Type, Type, Type, std::array<Type, 3>, Type, Type, Type>>().template extra<11, Values...>(2, std::declval<Misc>(...)), class = typename std::enable_if<std::is_void<Template>::value>::type> Template chi (Misc && misc)

Template [Chi](#) (Misc &...Misc)

update <unsigned int> Vol;

- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<const ImagineA::AbstractStep<Photot<-Type, Dimension>, unsigned int, std::array<Type, 1+(1+Dimension)*2>, std::tuple<Type, Type, Type, Type, std::array<Type, Dimension>, Type, Type, Type, Type, Type, std::array<Type, 3>, Type, Type, Type>>().template extra<11, Values...>(2, std::declval<Misc>(...)), class = typename std::enable_if<std::is_void<Template>::value>::type>

Template chi (Misc &&...misc) const

Immutable access to the chi data.

- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<magrathea::AbstractStep<Photon<<-Type, Dimension>, unsigned int, std::array<Type, 1+(1+Dimension)*2>, std::tuple<Type, Type, Type, Type, std::array<Type, Dimension>, Type, Type, Type, Type, Type, Type, std::array<Type, 3>, Type, Type, Type>>>().template extra<12, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<!std::is_void<Template>::value>::type>

Template Lambda (Misc &&...misc)

Access to the lambda data.

- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<const magrathea::AbstractStep<Photon<-Type, Dimension>, unsigned int, std::array<Type, 1+(1+Dimension)*2>, std::tuple<Type, Type, Type, Type, std::array<Type, Dimension>, Type, Type, Type, Type, Type, std::array<Type, 3>, Type, Type, Type>>>().template extra<12, Values...>(std::declval<Misc>(...)), class = typename std::enable_if<std::is_void<Template>::value>::type>

Template Lambda (Misc &...misc) const

Immutable access to the lambda data.

- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<magrathea::AbstractStep<Photon<->Type, Dimension>, unsigned int, std::array<Type, 1+(1+Dimension)*2>, std::tuple<Type, Type, Type, Type, std::array<Type, Dimension>, Type, Type, Type, Type, Type, Type, std::array<Type, 3>, Type, Type, Type>>()).template extra<13, Values...>(std::declval<Misc>(),), class = typename std::enable_if<std::is_void<Template>::value>::type>

Template [dphidt](#) (Misc & misc)

Access to the dphidt data

- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<const magrathea::AbstractStep<Photon->::Type, Dimension>, std::array<Type, 1+(1+Dimension)*2>, std::tuple<Type, Type, Type, Type, std::array<Type, Dimension>, Type, Type, Type, Type, Type, std::array<Type, 3>, std::tuple<Type, Type, Type> >>().template extra<13, Values...>(std::declval<Misc>(), class... typeparams std::enable_if<Template::value> typeparams)

Template [dpbhidt](#) (Misc 8.8 - misc) const

Template **uplink** (Misc &&...Misc) contains table with highlighted

- template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<magrathea::AbstractStep<Photon<-Type, Dimension>, unsigned int, std::array<Type, 1+(1+Dimension)*2>, std::tuple<Type, Type, Type, Type, std::array<Type, Dimension>, Type, Type, Type, Type, Type, std::array<Type, 3>, Type, Type, Type>>>().template extra<14, Values...>(std-

::declval<Misc>(...)), class = type

Template **s** (Misc &&...)

- Access to the s data.*

 - template<unsigned int... Values, class... Misc, class Template = decltype(std::declval<const magrathea::AbstractStep<Photon<-Type, Dimension>, unsigned int, std::array<Type, 1+(1+Dimension)*2>, std::tuple<Type, Type, Type, Type, std::array<Type, Dimension>, Type, Type, Type, Type, Type, Type, std::array<Type, 3>, Type, Type, Type>>()).template extra<14, Values...>(std-

::declval<Misc>(...)), class = typename st

Immutable access to the s data.

Static Public Member Functions

Test

- static int `example ()`
Example function.

Public Attributes

- using `operator =` = `typedef`

Additional Inherited Members

6.40.1 Detailed Description

`template<typename Type = double, unsigned int Dimension = 3>exception Photon< Type, Dimension >`

`Photon` step implementation for raytracing.

A step of integration for a given photon.

Template Parameters

<code>Type</code>	Data type.
<code>Dimension</code>	Number of space dimension.

6.40.2 Constructor & Destructor Documentation

6.40.2.1 `template<typename Type , unsigned int Dimension> template<class... Misc> Photon< Type, Dimension >::Photon (Misc &&... misc) [inline], [explicit]`

Explicit generic constructor.

Provides a generic interface to all constructors of the base class.

Template Parameters

<code>Misc</code>	(Miscellaneous types.)
-------------------	------------------------

Parameters

<code>in</code>	<code>misc</code>	Miscellaneous arguments.
-----------------	-------------------	--------------------------

6.40.3 Member Function Documentation

6.40.3.1 `template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::a (Misc &&... misc) [inline]`

Access to the a data.

Provides an access to the a data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.2 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::a (Misc &&... misc) const [inline]

Immutable access to the a data.

Provides an immutable access to the a data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.3 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::ah (Misc &&... misc) [inline]

Access to the ah data.

Provides an access to the ah data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.4 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > **Template Photon< Type, Dimension >::ah** (*Misc &&... misc*) const [inline]

Immutable access to the ah data.

Provides an immutable access to the ah data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.5 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > **Template Photon< Type, Dimension >::chi** (*Misc &&... misc*) [inline]

Access to the chi data.

Provides an access to the chi data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.6 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > **Template Photon< Type, Dimension >::chi** (*Misc &&... misc*) const [inline]

Immutable access to the chi data.

Provides an immutable access to the chi data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.7 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::distance (Misc &&... misc) [inline]

Access to the distance data.

Provides an access to the distance data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.8 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::distance (Misc &&... misc) const [inline]

Immutable access to the distance data.

Provides an immutable access to the distance data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.9 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::dphidl (Misc &&... misc) [inline]

Access to the dphidl data.

Provides an access to the dphidl data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.10 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::dphidl (Misc &&... misc) const [inline]

Immutable access to the dphidl data.

Provides an immutable access to the dphidl data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.11 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::dphidt (Misc &&... misc) [inline]

Access to the dphidt data.

Provides an access to the dphidt data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.12 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::dphidt (Misc &&... misc) const [inline]

Immutable access to the dphidt data.

Provides an immutable access to the dphidt data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.13 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::dphidx (Misc &&... misc) [inline]

Access to the dphidx data.

Provides an access to the dphidx data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.14 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::dphidx (Misc &&... misc) const [inline]

Immutable access to the dphidx data.

Provides an immutable access to the dphidx data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.15 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::dphidy (Misc &&... *misc*) [inline]

Access to the dphidy data.

Provides an access to the dphidy data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.16 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::dphidy (Misc &&... *misc*) const [inline]

Immutable access to the dphidy data.

Provides an immutable access to the dphidy data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.17 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::dphidz (Misc &&... *misc*) [inline]

Access to the dphidz data.

Provides an access to the dphidz data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.18 `template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::dphidz(Misc &&... misc) const [inline]`

Immutable access to the dphidz data.

Provides an immutable access to the dphidz data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.19 `template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::dsdl2(Misc &&... misc) [inline]`

Access to the dsdl2 data.

Provides an access to the dsdl2 data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.20 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::dsdl2 (Misc &&... misc) const [inline]

Immutable access to the dsdl2 data.

Provides an immutable access to the dsdl2 data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.21 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::dtdl (Misc &&... misc) [inline]

Access to the dtdl data.

Provides an access to the dtdl data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.22 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::dtdl (Misc &&... misc) const [inline]

Immutable access to the dtdl data.

Provides an immutable access to the dtdl data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.23 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::dxdl (Misc &&... misc) [inline]

Access to the dxdl data.

Provides an access to the dxdl data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.24 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::dxdl (Misc &&... misc) const [inline]

Immutable access to the dxdl data.

Provides an immutable access to the dxdl data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.25 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::dydl (Misc &&... misc) [inline]

Access to the dydl data.

Provides an access to the dydl data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.26 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::dydl (Misc &&... misc) const [inline]

Immutable access to the dydl data.

Provides an immutable access to the dydl data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.27 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::dzdl (Misc &&... misc) [inline]

Access to the dzdl data.

Provides an access to the dzdl data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.28 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::dzdl (Misc &&... misc) const [inline]

Immutable access to the dzdl data.

Provides an immutable access to the dzdl data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.29 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::error (Misc &&... misc) [inline]

Access to the error data.

Provides an access to the error data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.30 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::error (Misc &&... misc) const [inline]

Immutable access to the error data.

Provides an immutable access to the error data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.31 template<typename Type , unsigned int Dimension> int Photon< Type, Dimension >::example() [static]

Example function.

Tests and demonstrates the use of Photon.

Returns

0 if no error.

6.40.3.32 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::index(Misc &&... misc) [inline]

Access to the index data.

Provides an access to the index data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.33 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::index(Misc &&... misc) const [inline]

Immutable access to the index data.

Provides an immutable access to the index data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.34 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon<Type, Dimension >::isw (Misc &&... misc) [inline]

Access to the isw data.

Provides an access to the isw data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.35 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon<Type, Dimension >::isw (Misc &&... misc) const [inline]

Immutable access to the isw data.

Provides an immutable access to the isw data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.36 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon<Type, Dimension >::iswold (Misc &&... misc) [inline]

Access to the iswold data.

Provides an access to the iswold data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.37 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::iswold(Misc &&... misc) const [inline]

Immutable access to the iswold data.

Provides an immutable access to the iswold data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.38 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::lambda(Misc &&... misc) [inline]

Access to the lambda data.

Provides an access to the lambda data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.39 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::lambda (Misc &&... misc) const [inline]

Immutable access to the lambda data.

Provides an immutable access to the lambda data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.40 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::laplacian (Misc &&... misc) [inline]

Access to the laplacian data.

Provides an access to the laplacian data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.41 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::laplacian (Misc &&... misc) const [inline]

Immutable access to the laplacian data.

Provides an immutable access to the laplacian data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.42 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::level (Misc &&... misc) [inline]

Access to the level data.

Provides an access to the level data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.43 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::level (Misc &&... misc) const [inline]

Immutable access to the level data.

Provides an immutable access to the level data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.44 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::phi (Misc &&... misc) [inline]

Access to the phi data.

Provides an access to the phi data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.45 `template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::phi (Misc &&... misc) const [inline]`

Immutable access to the phi data.

Provides an immutable access to the phi data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.46 `template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::redshift (Misc &&... misc) [inline]`

Access to the redshift data.

Provides an access to the redshift data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.47 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::redshift (Misc &&... misc) const [inline]

Immutable access to the redshift data.

Provides an immutable access to the redshift data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.48 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::rho (Misc &&... misc) [inline]

Access to the rho data.

Provides an access to the rho data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.49 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::rho (Misc &&... misc) const [inline]

Immutable access to the rho data.

Provides an immutable access to the rho data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.50 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::s (Misc &&... *misc*) [inline]

Access to the s data.

Provides an access to the s data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.51 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::s (Misc &&... *misc*) const [inline]

Immutable access to the s data.

Provides an immutable access to the s data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.52 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::t (Misc &&... *misc*) [inline]

Access to the t data.

Provides an access to the t data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.53 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::t(Misc &&... misc) const [inline]

Immutable access to the t data.

Provides an immutable access to the t data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.54 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::x(Misc &&... misc) [inline]

Access to the x data.

Provides an access to the x data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.55 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::x (*Misc &&... misc*) const [inline]

Immutable access to the x data.

Provides an immutable access to the x data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.56 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::y (*Misc &&... misc*) [inline]

Access to the y data.

Provides an access to the y data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.57 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::y (*Misc &&... misc*) const [inline]

Immutable access to the y data.

Provides an immutable access to the y data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.58 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::z(Misc &&... misc) [inline]

Access to the z data.

Provides an access to the z data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.3.59 template<typename Type , unsigned int Dimension> template<unsigned int... Values, class... Misc, class Template , class > Template Photon< Type, Dimension >::z(Misc &&... misc) const [inline]

Immutable access to the z data.

Provides an immutable access to the z data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.40.4 Member Data Documentation

6.40.4.1 template<typename Type = double, unsigned int Dimension = 3> using Photon< Type, Dimension >::operator =

The documentation for this exception was generated from the following file:

- /data/home/mbreton/magrathea_pathfinder/src/photon.h

6.41 Rays Class Reference

```
#include <rays.h>
```

Static Public Member Functions

- template<class Parameters , class Map >
static void **ReadParamFile** (Parameters &**parameters**, Map &**parameter**)
Read parameter file.

6.41.1 Member Function Documentation

6.41.1.1 template<class Parameters , class Map > void Rays::ReadParamFile (Parameters & *parameters*, Map & *parameter*) [static]

Read parameter file.

Read and put in a structure the parameters.

Template Parameters

<i>Parameters</i>	structure type
<i>Map</i>	map type

Parameters

<i>in,out</i>	<i>parameters</i>	Structure containing the parameters.
<i>in</i>	<i>parameter</i>	Contains parameters to be rewritten

The documentation for this class was generated from the following file:

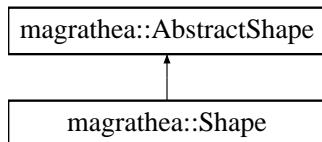
- /data/home/mbreton/magrathea_pathfinder/src/rays.h

6.42 magrathea::Shape Exception Reference

Basic implementation of shape.

```
#include <shape.h>
```

Inheritance diagram for magrathea::Shape:



Static Public Member Functions

Test

- static int [example \(\)](#)
Example function.

Additional Inherited Members

6.42.1 Detailed Description

Basic implementation of shape.

This class is the direct derivation of [AbstractShape](#). It provides the most basic and generic shape object without adding new functionalities to the abstract class.

6.42.2 Member Function Documentation

6.42.2.1 int magrathea::Shape::example () [static]

Example function.

Tests and demonstrates the use of [Shape](#).

Returns

0 if no error.

The documentation for this exception was generated from the following file:

- /data/home/mbreton/magrathea_pathfinder/src/magrathea/[shape.h](#)

6.43 magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > Exception Template Reference

A simple hyperoctree based on bit manipulations.

```
#include <simplehyperoctree.h>
```

Public Member Functions

Lifecycle

- [SimpleHyperOctree \(\)](#)
Implicit empty constructor.
- [SimpleHyperOctree \(const unsigned int ilvl, const unsigned int nref=0\)](#)
Explicit level constructor.

Operators

- bool [operator== \(const SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > &rhs\) const](#)
Equal to.
- bool [operator!= \(const SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > &rhs\) const](#)
Not equal to.
- Element & [operator\[\] \(const unsigned long long int ielem\)](#)
Element access operator.
- const Element & [operator\[\] \(const unsigned long long int ielem\) const](#)

- *Container & operator() ()*
Container access operator.
- *const Container & operator() () const*
Immutable container access operator.
- *Element & operator() (const Index &idx)*
Element access operator from hyperoctree index.
- *const Element & operator() (const Index &idx) const*
Immutable element access operator from hyperoctree index.
- *template<typename... Types, class = typename std::enable_if<(sizeof...(Types) != 0)>::type, class = typename std::enable_if<((std::is_convertible<typename std::tuple_element<0, std::tuple<typename std::remove_cv<typename std::remove_reference<-Types>::type>::type...> ::type, Type>::value) ? (sizeof...(Types) == Dimension) : (sizeof...(Types) == 1)) && (!std::is_same<typename std::tuple_element<0, std::tuple<typename std::remove_cv<typename std::remove_reference<Types>::type>::type...> ::type, Index>::value)>::type>*
Element & operator() (Types &&...iposs)
Element access operator from position.
- *template<typename... Types, class = typename std::enable_if<(sizeof...(Types) != 0)>::type, class = typename std::enable_if<((std::is_convertible<typename std::tuple_element<0, std::tuple<typename std::remove_cv<typename std::remove_reference<-Types>::type>::type...> ::type, Type>::value) ? (sizeof...(Types) == Dimension) : (sizeof...(Types) == 1)) && (!std::is_same<typename std::tuple_element<0, std::tuple<typename std::remove_cv<typename std::remove_reference<Types>::type>::type...> ::type, Index>::value)>::type>*
const Element & operator() (Types &&...iposs) const
Immutable element access operator from position.

Assignment

- *SimpleHyperOctree< Type, Index,
Data, Dimension, Position,
Extent, Element, Container > & assign ()*
Empty assignment.
- *SimpleHyperOctree< Type, Index,
Data, Dimension, Position,
Extent, Element, Container > & assign (const unsigned int ilvl, const unsigned int nref=0)*
Level assignment .
- *SimpleHyperOctree< Type, Index,
Data, Dimension, Position,
Extent, Element, Container > & assign (const SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > &source)*
Copy assignment .

Management

- *SimpleHyperOctree< Type, Index,
Data, Dimension, Position,
Extent, Element, Container > & nullify ()*
Nullify.
- *SimpleHyperOctree< Type, Index,
Data, Dimension, Position,
Extent, Element, Container > copy () const*
Copy.
- *template<class Template = SimpleHyperOctree<Type, Index, Data, Dimension, Position, Extent, Element, Container>, class = typename std::enable_if<std::is_constructible<Template, SimpleHyperOctree<Type, Index, Data, Dimension, Position, Extent, Element, Container> >::value>::type>*
Template cast () const
Cast.

Access

- *Element & at (const unsigned long long int ielem)*

- Access to an element with range-check.
- const Element & **at** (const unsigned long long int ielem) const
 - Immutable access to an element with range-check.*
- Element & **front** (const unsigned long long int ielem=0)
 - Access to the i-th element from the beginning.*
- const Element & **front** (const unsigned long long int ielem=0) const
 - Immutable access to the i-th element from the beginning.*
- Element & **back** (const unsigned long long int ielem=0)
 - Access to the i-th element from the end.*
- const Element & **back** (const unsigned long long int ielem=0) const
 - Immutable access to the i-th element from the end.*
- Element & **cycle** (const long long int ielem)
 - Cyclic access to an element.*
- const Element & **cycle** (const long long int ielem) const
 - Immutable cyclic access to an element.*
- Container & **container** ()
 - Direct access to the underlying container.*
- const Container & **container** () const
 - Direct access to the underlying container.*
- Element * **data** ()
 - Direct access to the underlying array.*
- const Element * **data** () const
 - Immutable direct access to the underlying array.*

Iterators

- template<typename Iterator = decltype(std::declval<Container>().begin()), class = typename std::enable_if<std::is_convertible<typename std::remove_cv<typename std::remove_reference<decltype(*std::declval<Iterator>())>::type>::type, Element>::value>::type>
 - Iterator begin ()**
 - Iterator to the beginning.*
- template<typename Iterator = decltype(std::declval<const Container>().begin()), class = typename std::enable_if<std::is_convertible<typename std::remove_cv<typename std::remove_reference<decltype(*std::declval<Iterator>())>::type>::type, Element>::value>::type>
 - Iterator begin () const**
 - Immutable iterator to the beginning.*
- template<typename Iterator = decltype(std::declval<Container>().cbegin()), class = typename std::enable_if<std::is_convertible<typename std::remove_cv<typename std::remove_reference<decltype(*std::declval<Iterator>())>::type>::type, Element>::value>::type>
 - Iterator cbegin () const**
 - Forced immutable iterator to the beginning.*
- template<typename Iterator = decltype(std::declval<Container>().end()), class = typename std::enable_if<std::is_convertible<typename std::remove_cv<typename std::remove_reference<decltype(*std::declval<Iterator>())>::type>::type, Element>::value>::type>
 - Iterator end ()**
 - Iterator to the end.*
- template<typename Iterator = decltype(std::declval<const Container>().end()), class = typename std::enable_if<std::is_convertible<typename std::remove_cv<typename std::remove_reference<decltype(*std::declval<Iterator>())>::type>::type, Element>::value>::type>
 - Iterator end () const**
 - Immutable iterator to the end.*
- template<typename Iterator = decltype(std::declval<Container>().cend()), class = typename std::enable_if<std::is_convertible<typename std::remove_cv<typename std::remove_reference<decltype(*std::declval<Iterator>())>::type>::type, Element>::value>::type>
 - Iterator cend () const**
 - Forced immutable iterator to the end.*
- template<typename Iterator = decltype(std::declval<Container>().rbegin()), class = typename std::enable_if<std::is_convertible<typename std::remove_cv<typename std::remove_reference<decltype(*std::declval<Iterator>())>::type>::type, Element>::value>::type>
 - Iterator rbegin ()**
 - Forced immutable iterator to the end.*

Reverse iterator to the beginning.

- template<typename Iterator = decltype(std::declval<const Container>().rbegin()), class = typename std::enable_if<std::is_convertible<typename std::remove_cv<typename std::remove_reference<decltype(*std::declval<Iterator>())>::type, Element>::value>::type> Iterator **rbegin** () const

Immutable reverse iterator to the beginning.

- template<typename Iterator = decltype(std::declval<Container>().crbegin()), class = typename std::enable_if<std::is_convertible<typename std::remove_cv<typename std::remove_reference<decltype(*std::declval<Iterator>())>::type, Element>::value>::type> Iterator **crbegin** () const

Forced immutable reverse iterator to the beginning.

- template<typename Iterator = decltype(std::declval<Container>().rend()), class = typename std::enable_if<std::is_convertible<typename std::remove_cv<typename std::remove_reference<decltype(*std::declval<Iterator>())>::type, Element>::value>::type> Iterator **rend** ()

Reverse iterator to the end.

- template<typename Iterator = decltype(std::declval<const Container>().rend()), class = typename std::enable_if<std::is_convertible<typename std::remove_cv<typename std::remove_reference<decltype(*std::declval<Iterator>())>::type, Element>::value>::type> Iterator **rend** () const

Immutable reverse iterator to the end.

- template<typename Iterator = decltype(std::declval<Container>().crend()), class = typename std::enable_if<std::is_convertible<typename std::remove_cv<typename std::remove_reference<decltype(*std::declval<Iterator>())>::type, Element>::value>::type> Iterator **crend** () const

Forced immutable reverse iterator to the end.

- template<typename Iterator , class = typename std::enable_if<std::is_convertible<typename std::remove_cv<typename std::remove_reference<decltype(*std::declval<Iterator>())>::type, Element>::value>::type> unsigned long long int **index** (const Iterator &it) const

Index of an iterator in the underlying container.

Search

- template<typename Iterator = decltype(std::declval<Container>().begin()), class = typename std::enable_if<std::is_convertible<typename std::remove_cv<typename std::remove_reference<decltype(*std::declval<Iterator>())>::type, Element>::value>::type> Iterator **find** (const Index &idx)
- template<typename Iterator = decltype(std::declval<const Container>().begin()), class = typename std::enable_if<std::is_convertible<typename std::remove_cv<typename std::remove_reference<decltype(*std::declval<Iterator>())>::type, Element>::value>::type> Iterator **find** (const Index &idx) const
- template<typename Iterator = decltype(std::declval<Container>().begin()), typename... Types, class = typename std::enable_if<(std::is_convertible<typename std::remove_cv<typename std::remove_reference<decltype(*std::declval<Iterator>())>::type, Element>::value) && ((std::is_convertible<typename std::tuple_element<0, std::tuple<typename std::remove_cv<typename std::remove_reference<Types>::type>::type...>::type, Type>::value) ? (sizeof...(Types) == Dimension) : (sizeof...(Types) == 1)) && (!std::is_same<typename std::tuple_element<0, std::tuple<typename std::remove_cv<typename std::remove_reference<Types>::type>::type...>::type, Index>::value)>::type> Iterator **locate** (Types &&...iposs)
- template<typename Iterator = decltype(std::declval<const Container>().begin()), typename... Types, class = typename std::enable_if<(std::is_convertible<typename std::remove_cv<typename std::remove_reference<decltype(*std::declval<Iterator>())>::type, Element>::value) && ((std::is_convertible<typename std::tuple_element<0, std::tuple<typename std::remove_cv<typename std::remove_reference<Types>::type>::type...>::type, Type>::value) ? (sizeof...(Types) == Dimension) : (sizeof...(Types) == 1)) && (!std::is_same<typename std::tuple_element<0, std::tuple<typename std::remove_cv<typename std::remove_reference<Types>::type>::type...>::type, Index>::value)>::type> Iterator **locate** (Types &&...iposs) const

Capacity

- bool **empty** () const

Emptiness checking.

- `unsigned long long int size () const`
Number of elements.
- `unsigned long long int capacity () const`
Capacity of the underlying storage.
- `unsigned long long int space () const`
Available space.
- `SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & shrink ()`
Shrink reserved storage.
- `SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & reserve (const unsigned long long int nelem)`
Increase reserved storage.

Modifiers

- `SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & fullclear ()`
Full clear the content.
- `SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & clear ()`
Clear the contents.
- `SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & pop ()`
Pop back.
- template<class... Misc>
`SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & append (Misc &&...misc)`
Append an element.
- template<class... Misc>
`SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & resize (Misc &&...misc)`
Resize.
- template<unsigned int... Values, class... Misc, class Template , class >
`SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & rho (Misc &&...misc)`
Access to the rho data.
- template<unsigned int... Values, class... Misc, class Template , class >
`SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & rho (Misc &&...misc) const`
Access to the rho data.
- template<unsigned int... Values, class... Misc, class Template , class >
`SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & phi (Misc &&...misc)`
Access to the phi data.
- template<unsigned int... Values, class... Misc, class Template , class >
`SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & phi (Misc &&...misc) const`
Immutable access to the phi data.

- template<unsigned int... Values, class... Misc, class Template , class >
SimpleHyperOctree< Type, Index,
Data, Dimension, Position,
Extent, Element, Container > & **dphidxyz** (Misc &&...misc)
Access to the dphidxyz data.
- template<unsigned int... Values, class... Misc, class Template , class >
SimpleHyperOctree< Type, Index,
Data, Dimension, Position,
Extent, Element, Container > & **dphidxyz** (Misc &&...misc) const
Immutable access to the dphidxyz data.
- template<unsigned int... Values, class... Misc, class Template , class >
SimpleHyperOctree< Type, Index,
Data, Dimension, Position,
Extent, Element, Container > & **dphidx** (Misc &&...misc)
Access to the dphidx data.
- template<unsigned int... Values, class... Misc, class Template , class >
SimpleHyperOctree< Type, Index,
Data, Dimension, Position,
Extent, Element, Container > & **dphidx** (Misc &&...misc) const
Immutable access to the dphidx data.
- template<unsigned int... Values, class... Misc, class Template , class >
SimpleHyperOctree< Type, Index,
Data, Dimension, Position,
Extent, Element, Container > & **dphidy** (Misc &&...misc)
Access to the dphidy data.
- template<unsigned int... Values, class... Misc, class Template , class >
SimpleHyperOctree< Type, Index,
Data, Dimension, Position,
Extent, Element, Container > & **dphidy** (Misc &&...misc) const
Immutable access to the dphidy data.
- template<unsigned int... Values, class... Misc, class Template , class >
SimpleHyperOctree< Type, Index,
Data, Dimension, Position,
Extent, Element, Container > & **dphidz** (Misc &&...misc)
Access to the dphidz data.
- template<unsigned int... Values, class... Misc, class Template , class >
SimpleHyperOctree< Type, Index,
Data, Dimension, Position,
Extent, Element, Container > & **dphidz** (Misc &&...misc) const
Immutable access to the dphidz data.
- template<unsigned int... Values, class... Misc, class Template , class >
SimpleHyperOctree< Type, Index,
Data, Dimension, Position,
Extent, Element, Container > & **a** (Misc &&...misc)
Access to the a data.
- template<unsigned int... Values, class... Misc, class Template , class >
SimpleHyperOctree< Type, Index,
Data, Dimension, Position,
Extent, Element, Container > & **a** (Misc &&...misc) const
Immutable access to the a data.

Refinement

- **SimpleHyperOctree**< Type, Index,
Data, Dimension, Position,
Extent, Element, Container > & **update ()**
Update refinement.

- template<typename Iterator , class = typename std::enable_if<std::is_convertible<typename std::remove_cv<typename std::remove_reference<decltype(*std::declval<Iterator>())>::type>::type, Element>::value>::type>
bool root (const Iterator &it)
Root level.
- template<typename Iterator , class = typename std::enable_if<std::is_convertible<typename std::remove_cv<typename std::remove_reference<decltype(*std::declval<Iterator>())>::type>::type, Element>::value>::type>
bool leaf (const Iterator &it)
Leaf level.
- template<typename Iterator , class = typename std::enable_if<std::is_convertible<typename std::remove_cv<typename std::remove_reference<decltype(*std::declval<Iterator>())>::type>::type, Element>::value>::type>
SimpleHyperOctree< Type, Index,
Data, Dimension, Position,
Extent, Element, Container > & **refine** (const Iterator &it)
Refine an element.
- template<typename Iterator , class = typename std::enable_if<std::is_convertible<typename std::remove_cv<typename std::remove_reference<decltype(*std::declval<Iterator>())>::type>::type, Element>::value>::type>
SimpleHyperOctree< Type, Index,
Data, Dimension, Position,
Extent, Element, Container > & **coarsen** (const Iterator &it)
Coarsen an element.

Interpolation

- template<typename... Types, class = typename std::enable_if<(sizeof...(Types) != 0)>::type, class = typename std::enable_if<((std::is_convertible<typename std::tuple_element<0, std::tuple<typename std::remove_cv<typename std::remove_reference<Types>::type>::type...> >::type, Type>::value) ? (sizeof...(Types) == Dimension) : (sizeof...(Types) == 1)) && (!std::is_same<typename std::tuple_element<0, std::tuple<typename std::remove_cv<typename std::remove_reference<Types>::type>::type...> >::type, Index>::value)>::type>
Data ngp (Types &&...iposs) const
Nearest grid point interpolation.
- template<typename... Types, class = typename std::enable_if<(sizeof...(Types) != 0)>::type, class = typename std::enable_if<((std::is_convertible<typename std::tuple_element<0, std::tuple<typename std::remove_cv<typename std::remove_reference<Types>::type>::type...> >::type, Type>::value) ? (sizeof...(Types) == Dimension) : (sizeof...(Types) == 1)) && (!std::is_same<typename std::tuple_element<0, std::tuple<typename std::remove_cv<typename std::remove_reference<Types>::type>::type...> >::type, Index>::value)>::type>
Data cic (Types &&...iposs) const
Cloud in cell interpolation.
- template<typename... Types, class = typename std::enable_if<(sizeof...(Types) != 0)>::type, class = typename std::enable_if<((std::is_convertible<typename std::tuple_element<0, std::tuple<typename std::remove_cv<typename std::remove_reference<Types>::type>::type...> >::type, Type>::value) ? (sizeof...(Types) == Dimension) : (sizeof...(Types) == 1)) && (!std::is_same<typename std::tuple_element<0, std::tuple<typename std::remove_cv<typename std::remove_reference<Types>::type>::type...> >::type, Index>::value)>::type>
Data tsc (Types &&...iposs) const
Triangular Shaped Cloud interpolation.
- template<typename... Types, class = typename std::enable_if<(sizeof...(Types) != 0)>::type, class = typename std::enable_if<((std::is_convertible<typename std::tuple_element<0, std::tuple<typename std::remove_cv<typename std::remove_reference<Types>::type>::type...> >::type, Type>::value) ? (sizeof...(Types) == Dimension) : (sizeof...(Types) == 1)) && (!std::is_same<typename std::tuple_element<0, std::tuple<typename std::remove_cv<typename std::remove_reference<Types>::type>::type...> >::type, Index>::value)>::type>
Data tsc (std::vector< Element > &elemsTsc, Types &&...iposs) const
Triangular Shaped Cloud interpolation.

Static Public Member Functions

Properties

- static constexpr Type **type** ()
Type.

- static constexpr Position **position** ()

Position.
- static constexpr Extent **extent** ()

Extent.
- static constexpr Element **element** ()

Element.
- static constexpr unsigned int **dimension** ()

Number of dimensions.

Friends

Stream

- template<typename SelfType , class SelfIndex , class SelfData , unsigned int SelfDimension, class SelfPosition , class SelfExtent , class SelfElement , class SelfContainer >
 std::ostream & **operator<<** (std::ostream &lhs, const SimpleHyperOctree< SelfType, SelfIndex, SelfData, SelfDimension, SelfPosition, SelfExtent, SelfElement, SelfContainer > &rhs)

Output stream operator.

6.43.1 Detailed Description

```
template<typename Type = double, class Index = SimpleHyperOctreeIndex<unsigned long long int, 3>, class Data = double,
unsigned int Dimension = Index::dimension(), class Position = std::ratio<0>, class Extent = std::ratio<1>, class Element = std-
::pair<Index, Data>, class Container = std::vector<Element>>exception magrathea::SimpleHyperOctree< Type, Index, Data,
Dimension, Position, Extent, Element, Container >
```

A simple hyperoctree based on bit manipulations.

Implementation of a simple and easy-to-use and hyperoctree structure in arbitrary dimension. It provides basic find and search algorithms based on indices relying on bit manipulations.

Template Parameters

<i>Type</i>	Scalar position type.
<i>Index</i>	Index type.
<i>Data</i>	Data type.
<i>Dimension</i>	Number of dimensions.
<i>Position</i>	Position of the hyperoctree center.
<i>Extent</i>	Extent of the hyperoctree.
<i>Element</i>	Underlying element type.
<i>Container</i>	Underlying container type.

6.43.2 Constructor & Destructor Documentation

6.43.2.1 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class
Element , class Container > magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent,
Element, Container >::SimpleHyperOctree() [inline]

Implicit empty constructor.

Provides an implicit construction of an empty hyperoctree.

6.43.2.2 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::SimpleHyperOctree (const unsigned int *ilvl*, const unsigned int *nref* = 0) [inline], [explicit]

Explicit level constructor.

Provides an explicit construction of a fixed mesh starting from the specified level and refining the provided amount of times.

Parameters

in	<i>ilvl</i>	Index of the first level.
in	<i>nref</i>	Number of refinements of the first level.

6.43.3 Member Function Documentation

6.43.3.1 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > template<unsigned int... Values, class... Misc, class Template , class > SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::a (Misc &... misc) [inline]

Access to the a data.

Provides an access to the a data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.43.3.2 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > template<unsigned int... Values, class... Misc, class Template , class > SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::a (Misc &... misc) const [inline]

Immutable access to the a data.

Provides an access to the a data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.43.3.3 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > template<class... Misc> SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::append (Misc &... *misc*) [inline]

Append an element.

Appends an element by constructing it from the arguments and emplacing it at the end of the hyperoctree.

Template Parameters

	<i>Misc</i>	(Miscellaneous types.)
--	-------------	------------------------

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Self reference.

6.43.3.4 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::assign () [inline]

Empty assignment.

Provides an empty assignment of an empty hyperoctree.

Returns

Self reference.

6.43.3.5 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::assign (const unsigned int *ilvl*, const unsigned int *nref* = 0) [inline]

Level assignment .

Provides an assignment of a fixed mesh starting from the specified level and refining the provided amount of times.

Parameters

in	<i>ilvl</i>	Index of the first level.
in	<i>nref</i>	Number of refinements of the first level.

Returns

Self reference.

6.43.3.6 template<typename Type , class Index , class Data , unsigned int Dimension , class Position , class Extent , class Element , class Container > SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::assign (const SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & source) [inline]

Copy assignment .

Provides a copy assignment from another hyperoctree.

Parameters

in	source	Source of the copy.
----	--------	---------------------

Returns

Self reference.

6.43.3.7 template<typename Type , class Index , class Data , unsigned int Dimension , class Position , class Extent , class Element , class Container > Element & magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::at (const unsigned long long int ielem) [inline]

Access to an element with range-check.

Provides an access to the specified element with a range-check.

Parameters

in	ielem	Index of the element in the underlying container.
----	-------	---

Returns

Reference to the element.

6.43.3.8 template<typename Type , class Index , class Data , unsigned int Dimension , class Position , class Extent , class Element , class Container > const Element & magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::at (const unsigned long long int ielem) const [inline]

Immutable access to an element with range-check.

Provides an access to the specified element with a range-check.

Parameters

in	ielem	Index of the element in the underlying container.
----	-------	---

Returns

Immutable reference to the element.

6.43.3.9 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > Element & magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::back (const unsigned long long int *ielem* = 0) [inline]

Access to the i-th element from the end.

Returns a reference to the i-th last element in the underlying container without doing any range check.

Parameters

in	<i>ielem</i>	Index of the element in the underlying container.
----	--------------	---

Returns

Reference to the element.

6.43.3.10 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > const Element & magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::back (const unsigned long long int *ielem* = 0) const [inline]

Immutable access to the i-th element from the end.

Returns a reference to the i-th last element in the underlying container without doing any range check.

Parameters

in	<i>ielem</i>	Index of the element in the underlying container.
----	--------------	---

Returns

Immutable reference to the element.

6.43.3.11 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > template<typename Iterator , class > Iterator magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::begin () [inline]

Iterator to the beginning.

Returns a pointer to the first element.

Template Parameters

Iterator	(Iterator type.)
----------	------------------

Returns

Pointer to the beginning.

6.43.3.12 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > template<typename Iterator , class > Iterator magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::begin () const [inline]

Immutable iterator to the beginning.

Returns a pointer to the first element.

Template Parameters

<i>Iterator</i>	(Iterator type.)
-----------------	------------------

Returns

Immutable pointer to the beginning.

6.43.3.13 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > unsigned long long int magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::capacity() const [inline]

Capacity of the underlying storage.

Returns the number of elements that the hyperoctree has currently allocated space for.

Returns

Capacity of the currently allocated underlying storage.

6.43.3.14 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > template<class Template , class > Template magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::cast() const [inline]

Cast.

Casts contents to another object type.

Template Parameters

<i>Template</i>	Output type.
-----------------	--------------

Returns

Casted copy.

6.43.3.15 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > template<typename Iterator , class > Iterator magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::cbegin() const [inline]

Forced immutable iterator to the beginning.

Returns a constant pointer to the first element.

Template Parameters

<i>Iterator</i>	(Iterator type.)
-----------------	------------------

Returns

Immutable pointer to the beginning.

6.43.3.16 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > template<typename Iterator , class > Iterator magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::cend() const [inline]

Forced immutable iterator to the end.

Returns a constant pointer to the position after the last element.

Template Parameters

<i>Iterator</i>	(Iterator type.)
-----------------	------------------

Returns

Immutable pointer to the end.

6.43.3.17 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > template<typename... Types, class , class > Data magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::cic(Types &&... *iposs*) const [inline]

Cloud in cell interpolation.

Computes the value of the data at the provided position using a cloud in cell interpolation scheme.

Template Parameters

<i>Types</i>	(Scalar position types.)
--------------	--------------------------

Parameters

<i>in</i>	<i>iposs</i>	Real positions along each dimension.
-----------	--------------	--------------------------------------

Returns

Value of the data at the provided position.

6.43.3.18 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::clear() [inline]

Clear the contents.

Removes all elements from the hyperoctree.

Returns

Self reference.

6.43.3.19 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > template<typename Iterator , class > SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::coarsen(const Iterator & it) [inline]

Coarsen an element.

Coarsens a given element by invalidating all the underlying children. To become valid again, the container should be updated.

Template Parameters

<i>Iterator</i>	(Iterator type.)
-----------------	------------------

Parameters

in	<i>it</i>	Iterator to an element.
----	-----------	-------------------------

Returns

Self reference.

6.43.3.20 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > Container & magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::container() [inline]

Direct access to the underlying container.

Provides a direct access to the underlying container by returning a reference to it.

Returns

Reference to the underlying container.

6.43.3.21 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > const Container & magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::container() const [inline]

Direct access to the underlying container.

Provides a direct access to the underlying container by returning a reference to it.

Returns

Immutable reference to the underlying container.

6.43.3.22 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::copy() const [inline]

Copy.

Generates a copy of the hyperoctree.

Returns

Copy.

6.43.3.23 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > template<typename Iterator , class > Iterator magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::crbegin() const [inline]

Forced immutable reverse iterator to the beginning.

Returns a constant reversed pointer to the position after the last element.

Template Parameters

<i>Iterator</i>	(Iterator type.)
-----------------	------------------

Returns

Immutable pointer to the end.

6.43.3.24 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > template<typename Iterator , class > Iterator magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::crend() const [inline]

Forced immutable reverse iterator to the end.

Returns a constant reversed pointer to the first element.

Template Parameters

<i>Iterator</i>	(Iterator type.)
-----------------	------------------

Returns

Immutable pointer to the beginning.

6.43.3.25 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > Element & magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::cycle(const long long int *ielem*) [inline]

Cyclic access to an element.

Provides a cyclic access to the elements, using the index modulo. Negative indexes are supported. It allows to iterate several times over the contents just by incrementing the provided index.

Parameters

in	<i>ielem</i>	Index of the element in the underlying container.
----	--------------	---

Returns

Reference to the element.

6.43.3.26 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > const Element & magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::cycle(const long long int *ielem*) const [inline]

Immutable cyclic access to an element.

Provides a cyclic access to the elements, using the index modulo. Negative indexes are supported. It allows to iterate several times over the contents just by incrementing the provided index.

Parameters

in	<i>ielem</i>	Index of the element in the underlying container.
----	--------------	---

Returns

Immutable reference to the element.

6.43.3.27 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > Element * magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::data() [inline]

Direct access to the underlying array.

Provides a direct access to the underlying array by returning a pointer to the first element of storage.

Returns

Pointer to the underlying element storage.

6.43.3.28 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > const Element * magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::data() const [inline]

Immutable direct access to the underlying array.

Provides a direct access to the underlying array by returning a pointer to the first element of storage.

Returns

Immutable pointer to the underlying element storage.

6.43.3.29 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > constexpr unsigned int magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::dimension() [static]

Number of dimensions.

Returns the number of spatial dimensions.

Returns

Copy of the number of dimensions.

6.43.3.30 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > template<unsigned int... Values, class... Misc, class Template , class > SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::dphidx(Misc &... misc) [inline]

Access to the dphidx data.

Provides an access to the dphidx data by forwarding parameters to the unified base accessor member.

Template Parameters

Values	List of template values.
Misc	(Miscellaneous types.)
Template	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

```
6.43.3.31 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent
, class Element , class Container > template<unsigned int... Values, class... Misc, class Template , class
> SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > &
magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container
>::dphidx ( Misc &&... misc ) const [inline]
```

Immutable access to the dphidx data.

Provides an access to the dphidx data by forwarding parameters to the unified base accessor member.

Template Parameters

Values	List of template values.
Misc	(Miscellaneous types.)
Template	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

```
6.43.3.32 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent
, class Element , class Container > template<unsigned int... Values, class... Misc, class Template , class
> SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > &
magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container
>::dphidxyz( Misc &&... misc ) [inline]
```

Access to the dphidxyz data.

Provides an access to the dphidxyz data by forwarding parameters to the unified base accessor member.

Template Parameters

Values	List of template values.
Misc	(Miscellaneous types.)
Template	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.43.3.33 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > template<unsigned int... Values, class... Misc, class Template , class > SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::dphidxyz(Misc &&... misc) const [inline]

Immutable access to the dphidxyz data.

Provides an access to the dphidxyz data by forwarding parameters to the unified base accessor member.

Template Parameters

Values	List of template values.
Misc	(Miscellaneous types.)
Template	(Deduced template type.)

Parameters

in	misc	Miscellaneous arguments.
----	------	--------------------------

Returns

Forwarded result.

6.43.3.34 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > template<unsigned int... Values, class... Misc, class Template , class > SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::dphidy(Misc &&... misc) [inline]

Access to the dphidy data.

Provides an access to the dphidy data by forwarding parameters to the unified base accessor member.

Template Parameters

Values	List of template values.
Misc	(Miscellaneous types.)
Template	(Deduced template type.)

Parameters

in	misc	Miscellaneous arguments.
----	------	--------------------------

Returns

Forwarded result.

6.43.3.35 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > template<unsigned int... Values, class... Misc, class Template , class > SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::dphidy(Misc &&... misc) const [inline]

Immutable access to the dphidy data.

Provides an access to the dphidy data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

```
6.43.3.36 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent
, class Element , class Container > template<unsigned int... Values, class... Misc, class Template , class
> SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > &
magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::dphidz
( Misc &&... misc ) [inline]
```

Access to the dphidz data.

Provides an access to the dphidz data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

```
6.43.3.37 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent
, class Element , class Container > template<unsigned int... Values, class... Misc, class Template , class
> SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > &
magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::dphidz
( Misc &&... misc ) const [inline]
```

Immutable access to the dphidz data.

Provides an access to the dphidz data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	misc	Miscellaneous arguments.
----	------	--------------------------

Returns

Forwarded result.

6.43.3.38 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > constexpr Element magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::element() [static]

Element.

Returns a copy of the default element value.

Returns

Copy of the default element value.

6.43.3.39 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > bool magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::empty() const [inline]

Emptiness checking.

Checks if the hyperoctree has no elements.

Returns

True if empty, false otherwise.

6.43.3.40 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > template<typename Iterator , class > Iterator magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::end() [inline]

Iterator to the end.

Returns a pointer to the position after the last element.

Template Parameters

Iterator	(Iterator type.)
----------	------------------

Returns

Pointer to the end.

6.43.3.41 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > template<typename Iterator , class > Iterator magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::end() const [inline]

Immutable iterator to the end.

Returns a pointer to the position after the last element.

Template Parameters

<i>Iterator</i>	(Iterator type.)
-----------------	------------------

Returns

Immutable pointer to the end.

6.43.3.42 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > constexpr Extent magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::extent () [static]

Extent.

Returns a copy of the default extent value.

Returns

Copy of the default extent value.

6.43.3.43 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > template<typename Iterator , class > Iterator magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::find (const Index & *idx*) [inline]

6.43.3.44 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > template<typename Iterator , class > Iterator magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::find (const Index & *idx*) const [inline]

6.43.3.45 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > Element & magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::front (const unsigned long long int *ielem* = 0) [inline]

Access to the i-th element from the beginning.

Returns a reference to the i-th first element in the underlying container without doing any range check.

Parameters

in	<i>ielem</i>	Index of the element in the underlying container.
----	--------------	---

Returns

Reference to the element.

6.43.3.46 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > const Element & magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::front (const unsigned long long int *ielem* = 0) const [inline]

Immutable access to the i-th element from the beginning.

Returns a reference to the i-th first element in the underlying container without doing any range check.

Parameters

in	<i>ielem</i>	Index of the element in the underlying container.
----	--------------	---

Returns

Immutable reference to the element.

6.43.3.47 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::fulclear() [inline]

Full clear the content.

Erase the hyperoctree.

Returns

Self reference.

6.43.3.48 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > template<typename Iterator , class > unsigned long long int magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::index(const Iterator & it) const [inline]

Index of an iterator in the underlying container.

Returns the index of the element pointed by an iterator.

Template Parameters

<i>Iterator</i>	(Iterator type.)
-----------------	------------------

Parameters

in	<i>it</i>	Iterator to an element.
----	-----------	-------------------------

Returns

Index of the element in the underlying container.

Exceptions

<i>std::out_of_range</i>	Out of range.
--------------------------	---------------

6.43.3.49 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > template<typename Iterator , class > bool magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::leaf(const Iterator & it) [inline]

Leaf level.

Checks whether the given element corresponds to a leaf level. If the element does not have any child, it is considered at the leaf level.

Template Parameters

<i>Iterator</i>	(Iterator type.)
-----------------	------------------

Parameters

in	<i>it</i>	Iterator to an element.
----	-----------	-------------------------

Returns

True if the element is not refined, false otherwise.

- 6.43.3.50 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > template<typename Iterator , typename... Types, class > Iterator **magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::locate** (Types &&... *iposs*) [inline]
- 6.43.3.51 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > template<typename Iterator , typename... Types, class > Iterator **magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::locate** (Types &&... *iposs*) const [inline]
- 6.43.3.52 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > template<typename... Types, class , class > Data **magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::ngp** (Types &&... *iposs*) const [inline]

Nearest grid point interpolation.

Computes the value of the data at the provided position using a nearest grid point interpolation scheme.

Template Parameters

Types	(Scalar position types.)
-------	--------------------------

Parameters

in	<i>iposs</i>	Real positions along each dimension.
----	--------------	--------------------------------------

Returns

Value of the data at the provided position.

- 6.43.3.53 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & **magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::nullify** () [inline]

Nullify.

Resets all data to their default values but keeping the same tree structure.

Returns

Self reference.

6.43.3.54 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > bool magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::operator!= (const SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & rhs) const [inline]

Not equal to.

Compares the hyperoctrees for difference.

Parameters

in	rhs	Right-hand side.
----	-----	------------------

Returns

True if not equal, false if equal.

6.43.3.55 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > Container & magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::operator()() [inline]

Container access operator.

Provides a direct access to the underlying container.

Returns

Reference to the container.

6.43.3.56 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > const Container & magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::operator()() const [inline]

Immutable container access operator.

Provides a direct access to the underlying container.

Returns

Immutable reference to the container.

6.43.3.57 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > Element & magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::operator()(const Index & idx) [inline]

Element access operator from hyperoctree index.

Provides a direct access to the element corresponding to the specified index.

Parameters

in	idx	Index value.
----	-----	--------------

Returns

Reference to the element.

6.43.3.58 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > const Element & magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::operator() (const Index & *idx*) const [inline]

Immutable element access operator from hyperoctree index.

Provides a direct access to the element corresponding to the specified index.

Parameters

in	<i>idx</i>	Index value.
----	------------	--------------

Returns

Immutable reference to the element.

6.43.3.59 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > template<typename... Types, class , class > Element & magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::operator() (Types &&... *iposs*) [inline]

Element access operator from position.

Provides a direct access to the element corresponding to the specified position.

Template Parameters

<i>Types</i>	(Scalar position types.)
--------------	--------------------------

Parameters

in	<i>iposs</i>	Real positions along each dimension.
----	--------------	--------------------------------------

Returns

Reference to the element.

6.43.3.60 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > template<typename... Types, class , class > const Element & magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::operator() (Types &&... *iposs*) const [inline]

Immutable element access operator from position.

Provides a direct access to the element corresponding to the specified position.

Template Parameters

<i>Types</i>	(Scalar position types.)
--------------	--------------------------

Parameters

in	<i>iposs</i>	Real positions along each dimension.
----	--------------	--------------------------------------

Returns

Immutable reference to the element.

6.43.3.61 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > bool magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::operator==(const SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & rhs) const [inline]

Equal to.

Compares the hyperoctrees for equality.

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

True if equal, false if not equal.

6.43.3.62 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > Element & magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::operator[](const unsigned long long int ielem) [inline]

Element access operator.

Provides a direct access to the specified element.

Parameters

in	<i>ielem</i>	Index of the element in the underlying container.
----	--------------	---

Returns

Reference to the element.

6.43.3.63 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > const Element & magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::operator[](const unsigned long long int ielem) const [inline]

Immutable element access operator.

Provides a direct access to the specified element.

Parameters

in	<i>ielem</i>	Index of the element in the underlying container.
----	--------------	---

Returns

Immutable reference to the element.

6.43.3.64 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > template<unsigned int... Values, class... Misc, class Template , class > SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::phi (Misc &&... misc) [inline]

Access to the phi data.

Provides an access to the phi data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.43.3.65 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > template<unsigned int... Values, class... Misc, class Template , class > SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::phi (Misc &&... misc) const [inline]

Immutable access to the phi data.

Provides an access to the phi data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

6.43.3.66 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::pop () [inline]

Pop back.

Removes the last element of the hyperoctree.

Returns

Self reference.

6.43.3.67 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > constexpr Position magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::position() [static]

Position.

Returns a copy of the default position value.

Returns

Copy of the default position value.

6.43.3.68 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > template<typename Iterator , class > Iterator magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::rbegin() [inline]

Reverse iterator to the beginning.

Returns a reversed pointer to the position after the last element.

Template Parameters

<i>Iterator</i>	(Iterator type.)
-----------------	------------------

Returns

Pointer to the end.

6.43.3.69 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > template<typename Iterator , class > Iterator magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::rbegin() const [inline]

Immutable reverse iterator to the beginning.

Returns a reversed pointer to the position after the last element.

Template Parameters

<i>Iterator</i>	(Iterator type.)
-----------------	------------------

Returns

Immutable pointer to the end.

6.43.3.70 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > template<typename Iterator , class > SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::refine(const Iterator & it) [inline]

Refine an element.

Refines a given element by inserting the related children at the end of the container. To become valid and accessible, the container should be updated.

Template Parameters

<i>Iterator</i>	(Iterator type.)
-----------------	------------------

Parameters

in	<i>it</i>	Iterator to an element.
----	-----------	-------------------------

Returns

Self reference.

6.43.3.71 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > template<typename Iterator , class > Iterator magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::rend() [inline]

Reverse iterator to the end.

Returns a reversed pointer to the first element.

Template Parameters

<i>Iterator</i>	(Iterator type.)
-----------------	------------------

Returns

Pointer to the beginning.

6.43.3.72 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > template<typename Iterator , class > Iterator magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::rend() const [inline]

Immutable reverse iterator to the end.

Returns a reversed pointer to the first element.

Template Parameters

<i>Iterator</i>	(Iterator type.)
-----------------	------------------

Returns

Immutable pointer to the beginning.

6.43.3.73 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::reserve(const unsigned long long int nelem) [inline]

Increase reserved storage.

Increases the capacity of the underlying storage. Existing elements are protected so it could not invalidate the actual contents.

Parameters

in	nelem	New reserved number of elements.
----	-------	----------------------------------

Returns

Self reference.

6.43.3.74 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > template<class... Misc> SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::resize (Misc &&... misc) [inline]

Resize.

Resizes the hyperoctree to contain a new number of elements.

Template Parameters

Misc	(Miscellaneous types.)
------	------------------------

Parameters

in	misc	Miscellaneous arguments.
----	------	--------------------------

Returns

Self reference.

6.43.3.75 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > template<unsigned int... Values, class... Misc, class Template , class > SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::rho (Misc &&... misc) [inline]

Access to the rho data.

Provides an access to the rho data by forwarding parameters to the unified base accessor member.

Template Parameters

Values	List of template values.
Misc	(Miscellaneous types.)
Template	(Deduced template type.)

Parameters

in	misc	Miscellaneous arguments.
----	------	--------------------------

Returns

Forwarded result.

```
6.43.3.76 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent
, class Element , class Container > template<unsigned int... Values, class... Misc, class Template , class
> SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > &
magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::rho (
Misc &&... misc ) const [inline]
```

Access to the rho data.

Provides an access to the rho data by forwarding parameters to the unified base accessor member.

Template Parameters

<i>Values</i>	List of template values.
<i>Misc</i>	(Miscellaneous types.)
<i>Template</i>	(Deduced template type.)

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Forwarded result.

```
6.43.3.77 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class
Element , class Container > template<typename Iterator , class > bool magrathea::SimpleHyperOctree<
Type, Index, Data, Dimension, Position, Extent, Element, Container >::root ( const Iterator & it ) [inline]
```

Root level.

Checks whether the given element corresponds to the root level.

Template Parameters

<i>Iterator</i>	(Iterator type.)
-----------------	------------------

Parameters

in	<i>it</i>	Iterator to an element.
----	-----------	-------------------------

Returns

True if the element is at the root level, false otherwise.

```
6.43.3.78 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class
Element , class Container > SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element,
Container > & magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element,
Container >::shrink ( ) [inline]
```

Shrink reserved storage.

Reduces memory usage by freeing unused memory.

Returns

Self reference.

6.43.3.79 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > unsigned long long int magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::size () const [inline]

Number of elements.

Returns the distance between the first and the last element.

Returns

The number of elements in the hyperoctree.

6.43.3.80 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > unsigned long long int magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::space () const [inline]

Available space.

Returns the maximum possible number of elements.

Returns

Maximum number of elements.

6.43.3.81 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > template<typename... Types, class , class > Data magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::tsc (Types &&... iposs) const [inline]

Triangular Shaped Cloud interpolation.

Computes the value of the data at the provided position using a triangular shaped cloud interpolation scheme.

Template Parameters

Types	(Scalar position types.)
-------	--------------------------

Parameters

in	iposs	Real positions along each dimension.
----	-------	--------------------------------------

Returns

Value of the data at the provided position.

6.43.3.82 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > template<typename... Types, class , class > Data magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::tsc (std::vector< Element > & elemsTsc, Types &&... iposs) const [inline]

Triangular Shaped Cloud interpolation.

Computes the value of the data at the provided position using a triangular shaped cloud interpolation scheme. Also use previsouly computed neighbouring cells

Template Parameters

Types	(Scalar position types.)
-------	--------------------------

Parameters

in	<i>elemsTsc</i>	Neighbouring cells previsouly computed
in	<i>iposs</i>	Real positions along each dimension.

Returns

Value of the data at the provided position.

6.43.3.83 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > constexpr Type magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::type () [static]

Type.

Returns a copy of the default type value.

Returns

Copy of the default type value.

6.43.3.84 template<typename Type , class Index , class Data , unsigned int Dimension, class Position , class Extent , class Element , class Container > SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container > & magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >::update () [inline]

Update refinement.

Updates the octree refinement by removing coarsened cells , by sorting refined ones and by removing cells with the same index.

Returns

Self reference.

6.43.4 Friends And Related Function Documentation

6.43.4.1 template<typename Type = double, class Index = SimpleHyperOctreeIndex<unsigned long long int, 3>, class Data = double, unsigned int Dimension = Index::dimension(), class Position = std::ratio<0>, class Extent = std::ratio<1>, class Element = std::pair<Index, Data>, class Container = std::vector<Element>> template<typename SelfType , class SelfIndex , class SelfData , unsigned int SelfDimension, class SelfPosition , class SelfExtent , class SelfElement , class SelfContainer > std::ostream& operator<< (std::ostream & lhs, const SimpleHyperOctree< SelfType, SelfIndex, SelfData, SelfDimension, SelfPosition, SelfExtent, SelfElement, SelfContainer > & rhs) [friend]

Output stream operator.

Displays the whole structure of the octree.

Template Parameters

<i>SelfType</i>	(Scalar position type.)
<i>SelfIndex</i>	(Index type.)
<i>SelfData</i>	(Data type.)
<i>SelfDimension</i>	(Number of dimensions.)
<i>SelfPosition</i>	(Position of the hyperoctree center.)
<i>SelfExtent</i>	(Extent of the hyperoctree.)
<i>SelfElement</i>	(Underlying element type.)
<i>SelfContainer</i>	(Underlying container type.)

Parameters

<i>in, out</i>	<i>lhs</i>	Left-hand side stream.
<i>in</i>	<i>rhs</i>	Right-hand side object.

Returns

Output stream.

The documentation for this exception was generated from the following file:

- /data/home/mbreton/magrathea_pathfinder/src/magrathea/simplehyperoctree.h

6.44 magrathea::SimpleHyperOctreeIndex< Type, Dimension, Bits > Exception Template Reference

A simple hyperoctree index based on an integer.

```
#include <simplehyperoctreeindex.h>
```

Public Member Functions

Lifecycle

- [SimpleHyperOctreeIndex \(\)](#)
Implicit empty constructor.
- [SimpleHyperOctreeIndex \(const Type source\)](#)
Explicit value constructor.
- template<class String , class = typename std::enable_if<(std::is_convertible<String, std::string>::value) && (std::is_convertible<decltype(std::declval<String>()[0]), char>::value)>::type>
[SimpleHyperOctreeIndex \(const String &source\)](#)
Explicit string constructor.

Operators

- [SimpleHyperOctreeIndex< Type, Dimension, Bits > & operator= \(const Type rhs\)](#)
Value assignment operator.
- template<class String , class = typename std::enable_if<(std::is_convertible<String, std::string>::value) && (std::is_convertible<decltype(std::declval<String>()[0]), char>::value)>::type>
[SimpleHyperOctreeIndex< Type, Dimension, Bits > & operator= \(const String &rhs\)](#)
String assignment operator.
- [operator Type \(\) const](#)
Implicit cast operator.

- Type & `operator()` ()

Data access operator.
- const Type & `operator()` () const

Immutable data access operator.
- bool `operator[]` (const unsigned int ibit) const

Immutable bit access.

Assignment

- `SimpleHyperOctreeIndex< Type, Dimension, Bits > & assign ()`

Empty assignment.
- `SimpleHyperOctreeIndex< Type, Dimension, Bits > & assign (const SimpleHyperOctreeIndex< Type, Dimension, Bits > &source)`

Copy assignment.
- `SimpleHyperOctreeIndex< Type, Dimension, Bits > & assign (const Type source)`

Value assignment.
- template<class String , class = typename std::enable_if<(std::is_convertible<String, std::string>::value) && (std::is_convertible<decltype(std::declval<String>()[0]), char>::value)>::type>
`SimpleHyperOctreeIndex< Type, Dimension, Bits > & assign (const String &source)`

String assignment.

Management

- Type & `data ()`

Data access.
- const Type & `data () const`

Immutable data access.
- Type `get () const`

Data getter.
- `SimpleHyperOctreeIndex< Type, Dimension, Bits > & set (const Type value)`

Data setter.
- `SimpleHyperOctreeIndex< Type, Dimension, Bits > & nullify ()`

Nullify.
- `SimpleHyperOctreeIndex< Type, Dimension, Bits > copy () const`

Copy.
- template<class OtherType = SimpleHyperOctreeIndex<Type, Dimension, Bits>, class = typename std::enable_if<std::is_constructible<OtherType, Type>::value>::type>
`OtherType cast () const`

Cast.
- template<typename Base = std::true_type, class = typename std::enable_if<((std::is_same<Base, std::true_type>::value) || (std::is_convertible<Base, int>::value)) && (!std::is_floating_point<Base>::value)>::type>
`std::string stringify (const Base base=Base()) const`

Stringify.

Core

- unsigned int `level () const`

Level.
- bool `coarsest () const`

Coarsest level.
- bool `finest () const`

Finest level.

- bool `limited` () const
Limited.
- bool `check` () const
Check.
- bool `invalidated` () const
Invalidated.
- `SimpleHyperOctreeIndex< Type, Dimension, Bits >` & `fix` ()
Fix.
- `SimpleHyperOctreeIndex< Type, Dimension, Bits >` & `invalidate` ()
Invalidate.

Tree

- `SimpleHyperOctreeIndex< Type, Dimension, Bits >` `parent` () const
Parent index.
- `SimpleHyperOctreeIndex< Type, Dimension, Bits >` `child` (const unsigned int isite) const
Child index.
- `SimpleHyperOctreeIndex< Type, Dimension, Bits >` `brother` (const unsigned int isite) const
Brother index.
- `SimpleHyperOctreeIndex< Type, Dimension, Bits >` `preceding` () const
Preceding brother index.
- `SimpleHyperOctreeIndex< Type, Dimension, Bits >` `following` () const
Following brother index.
- `SimpleHyperOctreeIndex< Type, Dimension, Bits >` `previous` (const unsigned int ilvl=0, const unsigned int nref=Bits/(Dimension+1)) const
Previous index.
- `SimpleHyperOctreeIndex< Type, Dimension, Bits >` `next` (const unsigned int ilvl=0, const unsigned int nref=Bits/(Dimension+1)) const
Next index.

6.44.1 Detailed Description

```
template<typename Type = unsigned long long int, unsigned int Dimension = 3, unsigned int Bits = sizeof(Type)*std::numeric_limits<unsigned char>::digits> exception magrathea::SimpleHyperOctreeIndex< Type, Dimension, Bits >
```

A simple hyperoctree index based on an integer.

Implements a simple hyperoctree index with no dependency using a single integer and providing all standard operations to easily recover level and position.

Template Parameters

<code>Type</code>	Unsigned integer type.
<code>Dimension</code>	Number of dimensions.
<code>Bits</code>	Size of the type in bits.

6.44.2 Constructor & Destructor Documentation

6.44.2.1 `template<typename Type , unsigned int Dimension, unsigned int Bits> magrathea::SimpleHyperOctreeIndex< Type, Dimension, Bits >::SimpleHyperOctreeIndex() [inline]`

Implicit empty constructor.

Provides an implicit construction of the index initialized to its default value.

6.44.2.2 `template<typename Type , unsigned int Dimension, unsigned int Bits> magrathea::SimpleHyperOctreeIndex< Type, Dimension, Bits >::SimpleHyperOctreeIndex(const Type source) [inline], [explicit]`

Explicit value constructor.

Provides an explicit construction of the index initialized to a particular value.

Parameters

<code>in</code>	<code>source</code>	Source of the copy.
-----------------	---------------------	---------------------

6.44.2.3 `template<typename Type , unsigned int Dimension, unsigned int Bits> template<class String , class > magrathea::SimpleHyperOctreeIndex< Type, Dimension, Bits >::SimpleHyperOctreeIndex(const String & source) [inline], [explicit]`

Explicit string constructor.

Provide an explicit construction from a string of zeros and ones. The index is filled from the most significant bit.

Template Parameters

<code>String</code>	String type.
---------------------	--------------

Parameters

<code>in</code>	<code>source</code>	Source of the copy.
-----------------	---------------------	---------------------

6.44.3 Member Function Documentation

6.44.3.1 `template<typename Type , unsigned int Dimension, unsigned int Bits> SimpleHyperOctreeIndex< Type, Dimension, Bits > & magrathea::SimpleHyperOctreeIndex< Type, Dimension, Bits >::assign() [inline]`

Empty assignment.

Assigns contents from an index initialized to its default value.

Returns

Self reference.

6.44.3.2 `template<typename Type , unsigned int Dimension, unsigned int Bits> SimpleHyperOctreeIndex< Type, Dimension, Bits > & magrathea::SimpleHyperOctreeIndex< Type, Dimension, Bits >::assign(const SimpleHyperOctreeIndex< Type, Dimension, Bits > & source) [inline]`

Copy assignment.

Assigns contents from the same type of index.

Parameters

in	source	Source of the copy.
----	--------	---------------------

Returns

Self reference.

6.44.3.3 template<typename Type , unsigned int Dimension, unsigned int Bits> SimpleHyperOctreeIndex< Type, Dimension, Bits > & magrathea::SimpleHyperOctreeIndex< Type, Dimension, Bits >::assign (const Type source) [inline]

Value assignment.

Assigns contents from an index value.

Parameters

in	source	Source of the copy.
----	--------	---------------------

Returns

Self reference.

6.44.3.4 template<typename Type , unsigned int Dimension, unsigned int Bits> template<class String , class > SimpleHyperOctreeIndex< Type, Dimension, Bits > & magrathea::SimpleHyperOctreeIndex< Type, Dimension, Bits >::assign (const String & source) [inline]

String assignment.

Assigns contents from a string of zeros and ones. The index is filled from the most significant bit.

Template Parameters

	String	String type.
--	--------	--------------

Parameters

in	source	Source of the copy.
----	--------	---------------------

Returns

Self reference.

6.44.3.5 template<typename Type , unsigned int Dimension, unsigned int Bits> SimpleHyperOctreeIndex< Type, Dimension, Bits > magrathea::SimpleHyperOctreeIndex< Type, Dimension, Bits >::brother (const unsigned int isite) const [inline]

Brother index.

Computes the brother index in the tree, which is an index with the same final parent.

Parameters

in	isite	Site of the brother.
----	-------	----------------------

Returns

Index of the specified brother.

6.44.3.6 template<typename Type , unsigned int Dimension, unsigned int Bits> template<class OtherType , class > OtherType magrathea::SimpleHyperOctreeIndex< Type, Dimension, Bits >::cast() const [inline]

Cast.

Casts the index to another object type.

Template Parameters

<i>OtherType</i>	Other data type.
------------------	------------------

Returns

Casted copy.

6.44.3.7 template<typename Type , unsigned int Dimension, unsigned int Bits> bool magrathea::SimpleHyperOctreeIndex< Type, Dimension, Bits >::check() const [inline]

Check.

Checks that the underlying integer represents an index with no error.

Returns

True if no error, false otherwise.

6.44.3.8 template<typename Type , unsigned int Dimension, unsigned int Bits> SimpleHyperOctreeIndex< Type, Dimension, Bits > magrathea::SimpleHyperOctreeIndex< Type, Dimension, Bits >::child(const unsigned int isite) const [inline]

Child index.

Computes the child index in the tree.

Parameters

in	<i>isite</i>	Site of the child.
----	--------------	--------------------

Returns

Index of the specified child.

6.44.3.9 template<typename Type , unsigned int Dimension, unsigned int Bits> bool magrathea::SimpleHyperOctreeIndex< Type, Dimension, Bits >::coarsest() const [inline]

Coarsest level.

Checks whether the index corresponds to the coarsest, unrefined level.

Returns

True if coarsest level, false otherwise.

6.44.3.10 template<typename Type , unsigned int Dimension, unsigned int Bits> SimpleHyperOctreeIndex< Type, Dimension, Bits > magrathea::SimpleHyperOctreeIndex< Type, Dimension, Bits >::copy () const [inline]

Copy.

Generates a copy of the index.

Returns

Copy.

6.44.3.11 template<typename Type , unsigned int Dimension, unsigned int Bits> Type & magrathea::SimpleHyperOctreeIndex< Type, Dimension, Bits >::data () [inline]

Data access.

Provides direct access to internal data.

Returns

Reference to underlying data.

6.44.3.12 template<typename Type , unsigned int Dimension, unsigned int Bits> const Type & magrathea::SimpleHyperOctreeIndex< Type, Dimension, Bits >::data () const [inline]

Immutable data access.

Provides an immutable direct access to internal data.

Returns

Immutable reference to underlying data.

6.44.3.13 template<typename Type , unsigned int Dimension, unsigned int Bits> bool magrathea::SimpleHyperOctreeIndex< Type, Dimension, Bits >::finest () const [inline]

Finest level.

Checks whether the index corresponds to the finest, most refined level.

Returns

True if finest level, false otherwise.

6.44.3.14 template<typename Type , unsigned int Dimension, unsigned int Bits> SimpleHyperOctreeIndex< Type, Dimension, Bits > & magrathea::SimpleHyperOctreeIndex< Type, Dimension, Bits >::fix () [inline]

Fix.

Fixes the underlying integer if it does not represents a correct index. To do so, every bit set after the last correct level is cleared.

Returns

Self reference.

6.44.3.15 template<typename Type , unsigned int Dimension, unsigned int Bits> **SimpleHyperOctreeIndex**< Type, Dimension, Bits > **magrathea::SimpleHyperOctreeIndex**< Type, Dimension, Bits >::following () const [inline]

Following brother index.

Computes the index of the following brother in the tree, which is an index with the same final parent. A cyclic operation is performed if the last brother of this final parent has already been reached.

Returns

Index of the following brother.

6.44.3.16 template<typename Type , unsigned int Dimension, unsigned int Bits> Type **magrathea::SimpleHyperOctreeIndex**< Type, Dimension, Bits >::get () const [inline]

Data getter.

Returns a copy of the internal data.

Returns

Copy of the underlying data.

6.44.3.17 template<typename Type , unsigned int Dimension, unsigned int Bits> **SimpleHyperOctreeIndex**< Type, Dimension, Bits > & **magrathea::SimpleHyperOctreeIndex**< Type, Dimension, Bits >::invalidate () [inline]

Invalidate.

Invalidates the index.

Returns

Self reference.

6.44.3.18 template<typename Type , unsigned int Dimension, unsigned int Bits> bool **magrathea::SimpleHyperOctreeIndex**< Type, Dimension, Bits >::invalidated () const [inline]

Invalidated.

Checks whether the index is invalidated.

Returns

True if the index is invalidated, false otherwise.

6.44.3.19 template<typename Type , unsigned int Dimension, unsigned int Bits> unsigned int **magrathea::SimpleHyperOctreeIndex**< Type, Dimension, Bits >::level () const [inline]

Level.

Computes the level of refinement according to the number of trailing zeros.

Returns

Refinement level.

6.44.3.20 template<typename Type , unsigned int Dimension, unsigned int Bits> bool magrathea::SimpleHyperOctreeIndex< Type, Dimension, Bits >::limited() const [inline]

Limited.

Checks whether the index corresponds to the coarsest or finest level of refinement.

Returns

True if coarsest or finest level, false otherwise.

6.44.3.21 template<typename Type , unsigned int Dimension, unsigned int Bits> SimpleHyperOctreeIndex< Type, Dimension, Bits > magrathea::SimpleHyperOctreeIndex< Type, Dimension, Bits >::next(const unsigned int *ilvl* = 0, const unsigned int *nref* = Bits / (Dimension+1)) const [inline]

Next index.

Computes the next index in the tree from the specified level to the specified number of refinements. A cyclic operation is performed if the last index has been reached.

Parameters

in	<i>ilvl</i>	Index of the first level.
in	<i>nref</i>	Number of refinements of the first level.

Returns

Index of the next element.

6.44.3.22 template<typename Type , unsigned int Dimension, unsigned int Bits> SimpleHyperOctreeIndex< Type, Dimension, Bits > & magrathea::SimpleHyperOctreeIndex< Type, Dimension, Bits >::nullify() [inline]

Nullify.

Resets the internal data to its default value.

Returns

Self reference.

6.44.3.23 template<typename Type , unsigned int Dimension, unsigned int Bits> magrathea::SimpleHyperOctreeIndex< Type, Dimension, Bits >::operator Type() const [inline]

Implicit cast operator.

Implicitly converts the index to an integer.

Returns

Copy of the underlying data.

6.44.3.24 template<typename Type , unsigned int Dimension, unsigned int Bits> Type & magrathea::SimpleHyperOctreeIndex< Type, Dimension, Bits >::operator()() [inline]

Data access operator.

Provides direct access to internal data.

Returns

Reference to underlying data.

```
6.44.3.25 template<typename Type , unsigned int Dimension, unsigned int Bits> const Type &
magrathea::SimpleHyperOctreeIndex< Type, Dimension, Bits >::operator() ( ) const [inline]
```

Immutable data access operator.

Provides an immutable direct access to internal data.

Returns

Immutable reference to underlying data.

```
6.44.3.26 template<typename Type , unsigned int Dimension, unsigned int Bits> SimpleHyperOctreeIndex< Type,
Dimension, Bits > & magrathea::SimpleHyperOctreeIndex< Type, Dimension, Bits >::operator=( const Type
rhs ) [inline]
```

Value assignment operator.

Assigns data from a value.

Parameters

<code>in</code>	<code>rhs</code>	Right-hand side.
-----------------	------------------	------------------

Returns

Self reference.

```
6.44.3.27 template<typename Type , unsigned int Dimension, unsigned int Bits> template<class String , class >
SimpleHyperOctreeIndex< Type, Dimension, Bits > & magrathea::SimpleHyperOctreeIndex< Type,
Dimension, Bits >::operator=( const String & rhs ) [inline]
```

String assignment operator.

Assigns data from a string of zeros and ones. The index is filled from the most significant bit.

Template Parameters

<code>String</code>	String type.
---------------------	--------------

Parameters

<code>in</code>	<code>rhs</code>	Right-hand side.
-----------------	------------------	------------------

Returns

Self reference.

```
6.44.3.28 template<typename Type , unsigned int Dimension, unsigned int Bits> bool magrathea::Simple-
HyperOctreeIndex< Type, Dimension, Bits >::operator[] ( const unsigned int ibit ) const
[inline]
```

Immutable bit access.

Provides an immutable access to the i-th bit.

Parameters

in	<i>ibit</i>	Bit index.
----	-------------	------------

Returns

Copy of the i-th bit.

6.44.3.29 template<typename Type , unsigned int Dimension, unsigned int Bits> SimpleHyperOctreeIndex< Type, Dimension, Bits > magrathea::SimpleHyperOctreeIndex< Type, Dimension, Bits >::parent () const [inline]

Parent index.

Computes the parent index in the tree.

Returns

Index of the parent.

6.44.3.30 template<typename Type , unsigned int Dimension, unsigned int Bits> SimpleHyperOctreeIndex< Type, Dimension, Bits > magrathea::SimpleHyperOctreeIndex< Type, Dimension, Bits >::preceding () const [inline]

Preceding brother index.

Computes the index of the preceding brother in the tree, which is an index with the same final parent. A cyclic operation is performed if the first brother of this final parent has already been reached.

Returns

Index of the preceding brother.

6.44.3.31 template<typename Type , unsigned int Dimension, unsigned int Bits> SimpleHyperOctreeIndex< Type, Dimension, Bits > magrathea::SimpleHyperOctreeIndex< Type, Dimension, Bits >::previous (const unsigned int *lvl* = 0, const unsigned int *nref* = Bits / (Dimension+1)) const [inline]

Previous index.

Computes the previous index in the tree from the specified level to the specified number of refinements. A cyclic operation is performed if the first index has been reached.

Parameters

in	<i>lvl</i>	Index of the first level.
in	<i>nref</i>	Number of refinements of the first level.

Returns

Index of the previous element.

6.44.3.32 template<typename Type , unsigned int Dimension, unsigned int Bits> SimpleHyperOctreeIndex< Type, Dimension, Bits > & magrathea::SimpleHyperOctreeIndex< Type, Dimension, Bits >::set (const Type value) [inline]

Data setter.

Sets the internal data to the provided value.

Parameters

in	value	Input value.
----	-------	---------------------

Returns

Self reference.

6.44.3.33 template<typename Type , unsigned int Dimension, unsigned int Bits> template<typename Base , class > std::string magrathea::SimpleHyperOctreeIndex< Type, Dimension, Bits >::stringify (const Base base = Base ()) const

Stringify.

Converts the index to a string. If no base is specified, the standard display is used.

Template Parameters

Base	Integral base type.
------	---------------------

Parameters

in	base	Integral base value.
----	------	----------------------

Returns

String corresponding to the index.

The documentation for this exception was generated from the following file:

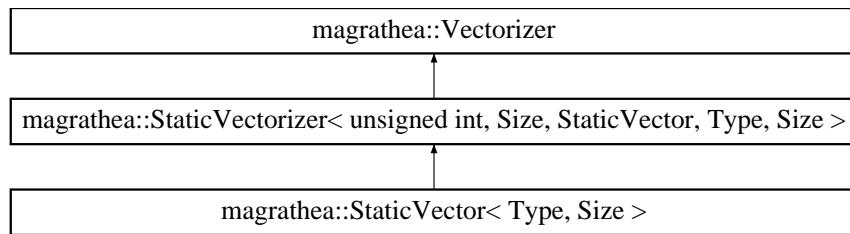
- /data/home/mbreton/magrathea_pathfinder/src/magrathea/simplehyperoctreeindex.h

6.45 magrathea::StaticVector< Type, Size > Exception Template Reference

Basic vectorized constant size container.

```
#include <staticvector.h>
```

Inheritance diagram for magrathea::StaticVector< Type, Size >:



Public Member Functions

Lifecycle

- **StaticVector ()**
Implicit empty constructor.
- template<typename FundamentalType = Type, class = typename std::enable_if<std::is_fundamental<FundamentalType>::value>::type>
StaticVector (const **StaticVector**< FundamentalType, Size > &source)
Implicit conversion constructor.
- template<typename OtherType = Type, class... Misc, class = typename std::enable_if<std::is_convertible<OtherType, Type>::value>::type>
StaticVector (const std::initializer_list< OtherType > &source, const Misc &...misc)
Implicit initializer list constructor.
- template<class... Misc, class = typename std::enable_if<sizeof...(Misc) != 0>::type>
StaticVector (const Misc &...misc)
Explicit generic constructor.

Vectorization

- Type & **operator[]** (const unsigned int i)
Direct access to the element.
- const Type & **operator[]** (const unsigned int i) const
Immutable direct access to the element.

Static Public Member Functions

Test

- static int **example** ()
Example function.

Public Attributes

- using **operator** = typedef

Protected Attributes

Data members

- Type **_data** [Size]
Data contents.

Additional Inherited Members

6.45.1 Detailed Description

`template<typename Type = double, unsigned int Size = 1>exception magrathea::StaticVector< Type, Size >`

Basic vectorized constant size container.

This class is the direct derivation of [StaticVectorizer](#). It provides the most basic vectorized constant size container without adding new functionalities to the abstract class.

Template Parameters

Type	Data type.
Size	Number of elements.

6.45.2 Constructor & Destructor Documentation

6.45.2.1 template<typename Type , unsigned int Size> magrathea::StaticVector< Type, Size >::StaticVector() [inline]

Implicit empty constructor.

Does nothing.

6.45.2.2 template<typename Type , unsigned int Size> template<typename FundamentalType , class > magrathea::StaticVector< Type, Size >::StaticVector(const StaticVector< FundamentalType, Size > & source) [inline]

Implicit conversion constructor.

Provides an implicit conversion from a fundamental type contents.

Template Parameters

FundamentalType	(Fundamental data type.)
-----------------	--------------------------

Parameters

in	source	Source of the copy.
----	--------	---------------------

6.45.2.3 template<typename Type , unsigned int Size> template<typename OtherType , class... Misc, class > magrathea::StaticVector< Type, Size >::StaticVector(const std::initializer_list< OtherType > & source, const Misc &... misc) [inline]

Implicit initializer list constructor.

Provides an implicit conversion from an initializer list.

Template Parameters

OtherType	(Other data type.)
Misc	(Miscellaneous types.)

Parameters

in	<i>source</i>	Source of the copy.
in	<i>misc</i>	Miscellaneous arguments.

6.45.2.4 template<typename Type , unsigned int Size> template<class... Misc, class > **magrathea::StaticVector< Type, Size >::StaticVector**(const Misc &... *misc*) [inline], [explicit]

Explicit generic constructor.

Provides a generic interface to all constructors of the base class. Before calling the associated constructor of the base class, the contents is initialized.

Template Parameters

<i>Misc</i>	(Miscellaneous types.)
-------------	------------------------

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

6.45.3 Member Function Documentation

6.45.3.1 template<typename Type , unsigned int Size> int **magrathea::StaticVector< Type, Size >::example**() [static]

Example function.

Tests and demonstrates the use of [StaticVector](#).

Returns

0 if no error.

6.45.3.2 template<typename Type , unsigned int Size> Type & **magrathea::StaticVector< Type, Size >::operator[]**(const unsigned int *i*) [inline]

Direct access to the element.

Provides a direct access to the specified element.

Parameters

in	<i>i</i>	Index of the element.
----	----------	-----------------------

Returns

Reference to the element.

6.45.3.3 template<typename Type , unsigned int Size> const Type & **magrathea::StaticVector< Type, Size >::operator[]**(const unsigned int *i*) const [inline]

Immutable direct access to the element.

Provides a constant direct access to the specified element.

Parameters

in	<i>i</i>	Index of the element.
----	----------	-----------------------

Returns

Const reference to the element.

6.45.4 Member Data Documentation

6.45.4.1 `template<typename Type = double, unsigned int Size = 1> Type magrathea::StaticVector< Type, Size >::data[Size] [protected]`

Data contents.

6.45.4.2 `template<typename Type = double, unsigned int Size = 1> using magrathea::StaticVector< Type, Size >::operator =`

The documentation for this exception was generated from the following file:

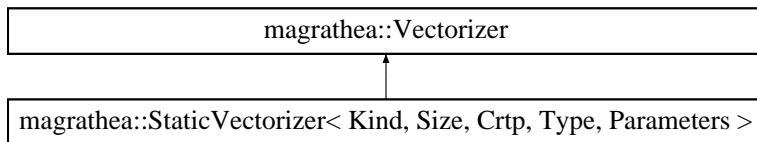
- /data/home/mbreton/magrathea_pathfinder/src/magrathea/[staticvector.h](#)

6.46 magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters > Class Template Reference

Helper class for generic constant size vectorization.

```
#include <staticvectorizer.h>
```

Inheritance diagram for magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >:



Public Member Functions

Vectorization

- `Type & operator[] (const unsigned int i)`
Direct access to the element.
- `const Type & operator[] (const unsigned int i) const`
Immutable direct access to the element.
- `Crtp< Type, Parameters...> & resize (const unsigned int n)`
Resize the container.

Operators : assignment

- `Crtp< Type, Parameters...> & operator= (const StaticVectorizer< Kind, Size, Crtp, Type, Parameters... > &rhs)`
Copy assignment operator.
- `template<typename OtherType , class = typename std::enable_if<std::is_convertible<OtherType, Type>::value>::type> Crtp< Type, Parameters...> & operator= (const std::initializer_list< OtherType > &rhs)`
Initializer list assignment operator.
- `template<class Misc > Crtp< Type, Parameters...> & operator= (const Misc &rhs)`
Copy assignment operator.

Operators : compound assignment

- template<class GenericType >
Crtp< Type, Parameters...> & **operator+=** (const GenericType &rhs)
Addition assignment.
- template<class GenericType >
Crtp< Type, Parameters...> & **operator-=** (const GenericType &rhs)
Subtraction assignment.
- template<class GenericType >
Crtp< Type, Parameters...> & **operator*=** (const GenericType &rhs)
Multiplication assignment.
- template<class GenericType >
Crtp< Type, Parameters...> & **operator/=** (const GenericType &rhs)
Division assignment.
- template<class GenericType >
Crtp< Type, Parameters...> & **operator%=** (const GenericType &rhs)
Modulo assignment.
- template<class GenericType >
Crtp< Type, Parameters...> & **operator&=** (const GenericType &rhs)
Bitwise AND assignment.
- template<class GenericType >
Crtp< Type, Parameters...> & **operator|=** (const GenericType &rhs)
Bitwise OR assignment.
- template<class GenericType >
Crtp< Type, Parameters...> & **operator^=** (const GenericType &rhs)
Bitwise XOR assignment.
- template<class GenericType >
Crtp< Type, Parameters...> & **operator<<=** (const GenericType &rhs)
Bitwise left shift assignment.
- template<class GenericType >
Crtp< Type, Parameters...> & **operator>>=** (const GenericType &rhs)
Bitwise right shift assignment.

Operators : main

- template<typename OtherType , class = typename std::enable_if<std::is_convertible<OtherType, Type>::value>::type>
Crtp< typename
std::common_type< Type,
OtherType >::type,
Parameters...> **operator+** (const **StaticVectorizer**< Kind, Size, Crtp, OtherType, Parameters...> &rhs)
const
Addition.
- template<typename OtherType , class = typename std::enable_if<std::is_convertible<OtherType, Type>::value>::type>
Crtp< typename
std::common_type< Type,
OtherType >::type,
Parameters...> **operator-** (const **StaticVectorizer**< Kind, Size, Crtp, OtherType, Parameters...> &rhs)
const
Subtraction.
- template<typename OtherType , class = typename std::enable_if<std::is_convertible<OtherType, Type>::value>::type>
Crtp< typename
std::common_type< Type,
OtherType >::type,
Parameters...> **operator*** (const **StaticVectorizer**< Kind, Size, Crtp, OtherType, Parameters...> &rhs)
const
Multiplication.

- template<typename OtherType , class = typename std::enable_if<std::is_convertible<OtherType, Type>::value>::type>
 $\text{Crtp} < \text{typename}$
 $\text{std::common_type}<\text{Type},$
 $\text{OtherType}>::\text{type}$,
 $\text{Parameters...}>$ **operator/** (const **StaticVectorizer**< Kind, Size, Crtp, OtherType, Parameters...> &rhs)
 const
Division.
- template<typename OtherType , class = typename std::enable_if<std::is_convertible<OtherType, Type>::value>::type>
 $\text{Crtp} < \text{typename}$
 $\text{std::common_type}<\text{Type},$
 $\text{OtherType}>::\text{type}$,
 $\text{Parameters...}>$ **operator%** (const **StaticVectorizer**< Kind, Size, Crtp, OtherType, Parameters...> &rhs)
 const
Modulo.
- template<typename OtherType , class = typename std::enable_if<std::is_convertible<OtherType, Type>::value>::type>
 $\text{Crtp} < \text{typename}$
 $\text{std::common_type}<\text{Type},$
 $\text{OtherType}>::\text{type}$,
 $\text{Parameters...}>$ **operator&** (const **StaticVectorizer**< Kind, Size, Crtp, OtherType, Parameters...> &rhs)
 const
Bitwise AND.
- template<typename OtherType , class = typename std::enable_if<std::is_convertible<OtherType, Type>::value>::type>
 $\text{Crtp} < \text{typename}$
 $\text{std::common_type}<\text{Type},$
 $\text{OtherType}>::\text{type}$,
 $\text{Parameters...}>$ **operator|** (const **StaticVectorizer**< Kind, Size, Crtp, OtherType, Parameters...> &rhs)
 const
Bitwise OR.
- template<typename OtherType , class = typename std::enable_if<std::is_convertible<OtherType, Type>::value>::type>
 $\text{Crtp} < \text{Type},$ Parameters...> **operator<<** (const **StaticVectorizer**< Kind, Size, Crtp, OtherType, Parameters...> &rhs) const
Bitwise left shift.
- template<typename OtherType , class = typename std::enable_if<std::is_convertible<OtherType, Type>::value>::type>
 $\text{Crtp} < \text{Type},$ Parameters...> **operator>>** (const **StaticVectorizer**< Kind, Size, Crtp, OtherType, Parameters...> &rhs) const
Bitwise right shift.
- template<typename OtherType , class = typename std::enable_if<std::is_convertible<OtherType, Type>::value>::type>
 $\text{Crtp} < \text{bool},$ Parameters...> **operator&&** (const **StaticVectorizer**< Kind, Size, Crtp, OtherType, Parameters...> &rhs) const
Logical AND.
- template<typename OtherType , class = typename std::enable_if<std::is_convertible<OtherType, Type>::value>::type>
 $\text{Crtp} < \text{bool},$ Parameters...> **operator||** (const **StaticVectorizer**< Kind, Size, Crtp, OtherType, Parameters...> &rhs) const
Logical OR.
- template<typename OtherType , class = typename std::enable_if<std::is_convertible<OtherType, Type>::value>::type>
 $\text{Crtp} < \text{bool},$ Parameters...> **operator==** (const **StaticVectorizer**< Kind, Size, Crtp, OtherType, Parameters...> &rhs) const
Equal to.
- template<typename OtherType , class = typename std::enable_if<std::is_convertible<OtherType, Type>::value>::type>
 $\text{Crtp} < \text{bool},$ Parameters...> **operator!=** (const **StaticVectorizer**< Kind, Size, Crtp, OtherType, Parameters...> &rhs) const
Not equal to.

- template<typename OtherType , class = typename std::enable_if<!std::is_convertible<OtherType, Type>::value>::type>
Crtp< bool, Parameters...> **operator>** (const **StaticVectorizer**< Kind, Size, Crtp, OtherType, Parameters...> &rhs) const
Greater than.
- template<typename OtherType , class = typename std::enable_if<!std::is_convertible<OtherType, Type>::value>::type>
Crtp< bool, Parameters...> **operator<** (const **StaticVectorizer**< Kind, Size, Crtp, OtherType, Parameters...> &rhs) const
Less than.
- template<typename OtherType , class = typename std::enable_if<!std::is_convertible<OtherType, Type>::value>::type>
Crtp< bool, Parameters...> **operator>=** (const **StaticVectorizer**< Kind, Size, Crtp, OtherType, Parameters...> &rhs) const
Greater than or equal to.
- template<typename OtherType , class = typename std::enable_if<!std::is_convertible<OtherType, Type>::value>::type>
Crtp< bool, Parameters...> **operator<=** (const **StaticVectorizer**< Kind, Size, Crtp, OtherType, Parameters...> &rhs) const
Less than or equal to.

Operators : with rhs value

- template<typename OtherType , class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, Type>::value)>::type>
Crtp< typename std::common_type< Type, OtherType >::type, Parameters...> **operator+** (const OtherType &rhs) const
Addition with rhs value.
- template<typename OtherType , class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, Type>::value)>::type>
Crtp< typename std::common_type< Type, OtherType >::type, Parameters...> **operator-** (const OtherType &rhs) const
Subtraction with rhs value.
- template<typename OtherType , class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, Type>::value)>::type>
Crtp< typename std::common_type< Type, OtherType >::type, Parameters...> **operator*** (const OtherType &rhs) const
Multiplication with rhs value.
- template<typename OtherType , class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, Type>::value)>::type>
Crtp< typename std::common_type< Type, OtherType >::type, Parameters...> **operator/** (const OtherType &rhs) const
Division with rhs value.
- template<typename OtherType , class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, Type>::value)>::type>
Crtp< typename std::common_type< Type, OtherType >::type, Parameters...> **operator%** (const OtherType &rhs) const
Modulo with rhs value.
- template<typename OtherType , class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, Type>::value)>::type>
Crtp< typename std::common_type< Type, OtherType >::type, Parameters...> **operator&** (const OtherType &rhs) const

- template<typename OtherType , class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, Type>::value)>::type>
Crtp< typename std::common_type< Type, OtherType >::type,
Parameters...> operator| (const OtherType &rhs) const

Bitwise AND with rhs value.
- template<typename OtherType , class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, Type>::value)>::type>
Crtp< typename std::common_type< Type, OtherType >::type,
Parameters...> operator^ (const OtherType &rhs) const

Bitwise OR with rhs value.
- template<typename OtherType , class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, Type>::value)>::type>
Crtp< Type, Parameters...> operator^^(const OtherType &rhs) const

Bitwise XOR with rhs value.
- template<typename OtherType , class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, Type>::value)>::type>
Crtp< Type, Parameters...> operator>> (const OtherType &rhs) const

Bitwise left shift with rhs value.
- template<typename OtherType , class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, Type>::value)>::type>
Crtp< Type, Parameters...> operator>>> (const OtherType &rhs) const

Bitwise right shift with rhs value.
- template<typename OtherType , class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, Type>::value)>::type>
Crtp< bool, Parameters...> operator&& (const OtherType &rhs) const

Logical AND with rhs value.
- template<typename OtherType , class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, Type>::value)>::type>
Crtp< bool, Parameters...> operator|| (const OtherType &rhs) const

Logical OR with rhs value.
- template<typename OtherType , class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, Type>::value)>::type>
Crtp< bool, Parameters...> operator== (const OtherType &rhs) const

Equal to with rhs value.
- template<typename OtherType , class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, Type>::value)>::type>
Crtp< bool, Parameters...> operator!= (const OtherType &rhs) const

Not equal to with rhs value.
- template<typename OtherType , class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, Type>::value)>::type>
Crtp< bool, Parameters...> operator> (const OtherType &rhs) const

Greater than with rhs value.
- template<typename OtherType , class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, Type>::value)>::type>
Crtp< bool, Parameters...> operator< (const OtherType &rhs) const

Less than with rhs value.
- template<typename OtherType , class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, Type>::value)>::type>
Crtp< bool, Parameters...> operator>= (const OtherType &rhs) const

Greater than or equal to with rhs value.
- template<typename OtherType , class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, Type>::value)>::type>
Crtp< bool, Parameters...> operator<= (const OtherType &rhs) const

Less than or equal to with rhs value.

Operators : unary

- **Crtp< bool, Parameters...> operator! () const**

Logical NOT.

- Crtp< Type, Parameters...> **operator~** () const
Bitwise NOT.
- Crtp< Type, Parameters...> **operator+** () const
Integer promotion.
- Crtp< Type, Parameters...> **operator-** () const
Additive inverse.
- Crtp< Type, Parameters...> & **operator++** ()
Increment prefix.
- Crtp< Type, Parameters...> & **operator--** ()
Decrement prefix.
- Crtp< Type, Parameters...> **operator++** (int)
Increment suffix.
- Crtp< Type, Parameters...> **operator--** (int)
Decrement suffix.

Access

- **StaticVectorizer< Kind, Size, Crtp, Type, Parameters...>** & **operator()** ()
Abstract class access.
- const **StaticVectorizer< Kind, Size, Crtp, Type, Parameters...>** & **operator()** () const
Immutable abstract class access.
- Type & **operator()** (const unsigned int i)
Monodimensional access operator.
- const Type & **operator()** (const unsigned int i) const
Immutable monodimensional access operator.
- Type & **at** (const unsigned int i)
Monodimensional access with range-check.
- const Type & **at** (const unsigned int i) const
Immutable monodimensional access with range-check.
- Type & **front** (const unsigned int i=0)
Monodimensional access to the i-th element from the beginning.
- const Type & **front** (const unsigned int i=0) const
Immutable monodimensional access to the i-th element from the beginning.
- Type & **back** (const unsigned int i=0)
Monodimensional access to the i-th element from the end.
- const Type & **back** (const unsigned int i=0) const
Immutable monodimensional access to the i-th element from the end.
- Type & **cycle** (const int i)
Cyclic monodimensional access to the contents.
- const Type & **cycle** (const int i) const
Immutable cyclic monodimensional access to the contents.

Assignment

- template<typename OtherType , class... Misc, class = typename std::enable_if<std::is_convertible<OtherType, Type>::value>-::type>
Crtp< Type, Parameters...> & **assign** (const std::initializer_list< OtherType > &source, const Misc &...misc)
Initializer list assignment.
- template<class... Misc>
Crtp< Type, Parameters...> & **assign** (const Misc &...misc)
Generic assignment.
- template<typename OtherType , class... Misc, class = typename std::enable_if<std::is_convertible<OtherType, Type>::value>-::type>
Crtp< Type, Parameters...> & **fill** (const std::initializer_list< OtherType > &source, const Misc &...misc)

- template<class... Misc>
`Crtp< Type, Parameters...> & fill (const Misc &...misc)`
Generic fill.
- template<typename OtherType , class... Misc, class = typename std::enable_if<std::is_convertible<OtherType, Type>::value>::type>
`Crtp< Type, Parameters...> replace (const std::initializer_list< OtherType > &source, const Misc &...misc)`
Initializer list replace.
- template<class... Misc>
`Crtp< Type, Parameters...> replace (const Misc &...misc)`
Generic replace.
- template<class GenericType >
`Crtp< Type, Parameters...> & put (const GenericType &source, const unsigned int pos, const unsigned int num=1)`
Put an element in the container.
- template<class GenericType >
`Crtp< Type, Parameters...> change (const GenericType &source, const unsigned int pos, const unsigned int num=1)`
Change an element of the container.

Management

- `Crtp< Type, Parameters...> & reserve (const unsigned int n)`
Reserve new space for the container.
- `Crtp< Type, Parameters...> & clear ()`
Clear contents.
- template<class... Location>
`Crtp< Type, Parameters...> & nullify (const Location &...location)`
Set elements to their default values.
- template<class... Location>
`Crtp< Type, Parameters...> & swap (StaticVectorizer< Kind, Size, Crtp, Type, Parameters...> &rhs, const Location &...location)`
Swap elements by copy.
- `Crtp< Type, Parameters...> copy () const`
Copy.
- template<typename OtherType = Type, class = typename std::enable_if<std::is_convertible<OtherType, Type>::value>::type>
`Crtp< OtherType, Parameters...> cast () const`
Cast to a different data type.

Comparison

- `bool null () const`
Check whether all elements are null.
- template<class GenericType >
`bool eq (const GenericType &rhs) const`
Compare for equality.
- template<class GenericType >
`bool ne (const GenericType &rhs) const`
Compare for difference.

Statistics

- template<class Mask = std::true_type, class = typename std::enable_if<(std::is_base_of<Vectorizer, Mask>::value) || (std::is_same<std::true_type, Mask>::value)>::type>
`const Type & min (const Mask &bitmask=Mask()) const`
Minimum element.

- template<class Mask = std::true_type, class = typename std::enable_if<(std::is_base_of<Vectorizer, Mask>::value) || (std::is_same<std::true_type, Mask>::value)>::type>
const Type & **max** (const Mask &bitmask=Mask()) const
Maximum element.

Application

- template<typename Return = Type, class Function , class... Args, class = typename std::enable_if<!std::is_base_of<Vectorizer, typename std::decay<Function>::type>::value>::type>
Crtp< Type, Parameters...> & **modify** (Function &&f, Args &&...args)
Modification by a function object.
- template<typename Return = Type, class Mask , class Function , class... Args, class = typename std::enable_if<(std::is_base_of<Vectorizer, Mask>::value) && (!std::is_base_of<Vectorizer, typename std::decay<Function>::type>::value)>::type>
Crtp< Type, Parameters...> & **modify** (const Mask &bitmask, Function &&f, Args &&...args)
Masked modification by a function object.
- template<typename Return = Type, class Function , class... Args, class = typename std::enable_if<!std::is_base_of<Vectorizer, typename std::decay<Function>::type>::value>::type>
Crtp< Return, Parameters...> **apply** (Function &&f, Args &&...args) const
Application of a function object.
- template<typename Return = Type, class Mask , class Function , class... Args, class = typename std::enable_if<(std::is_base_of<Vectorizer, Mask>::value) && (!std::is_base_of<Vectorizer, typename std::decay<Function>::type>::value)>::type>
Crtp< Return, Parameters...> **apply** (const Mask &bitmask, Function &&f, Args &&...args) const
Masked application of a function object.
- template<typename Return = Type, class Function = std::plus<Type>, class = typename std::enable_if<!std::is_base_of<Vectorizer, typename std::decay<Function>::type>::value>::type>
Return **reduce** (Function &&f=Function(), const Return &init=Return()) const
Reduction by a function object.
- template<typename Return = Type, class Mask , class Function = std::plus<Type>, class = typename std::enable_if<(std::is_base_of<Vectorizer, Mask>::value) && (!std::is_base_of<Vectorizer, typename std::decay<Function>::type>::value)>::type>
Return **reduce** (const Mask &bitmask, Function &&f=Function(), const Return &init=Return()) const
Reduction by a function object.
- template<typename Result = void, class Function , class Arg , class... Args, typename Return = typename std::conditional<std::is_void<Result>::value, typename std::common_type<Type, Arg, Args...>::type, Result>::type>
Crtp< Return, Parameters...> **combine** (Function &&f, const **StaticVectorizer**< Kind, Size, Crtp, Arg, Parameters...> &arg, const **StaticVectorizer**< Kind, Size, Crtp, Args, Parameters...> &...args) const
Combination by a function object.
- template<typename Return = Type, class Function >
Crtp< Return, Parameters...> **combine** (Function &&) const
Unique combination by a function object.

Count

- template<class Reference = bool, class Mask = std::true_type, class = typename std::enable_if<((std::is_convertible<Type, Reference>::value) || (std::is_base_of<Vectorizer, Reference>::value)) && ((std::is_base_of<Vectorizer, Mask>::value) || (std::is_same<std::true_type, Mask>::value))>::type>
unsigned int **count** (const Reference &r=true, const Mask &bitmask=Mask()) const
Count values.
- template<class Function , class Mask = std::true_type, class = typename std::enable_if<((!std::is_convertible<Type, typename std::decay<Function>::type>::value) && (!std::is_base_of<Vectorizer, typename std::decay<Function>::type>::value)) && ((std::is_base_of<Vectorizer, Mask>::value) || (std::is_same<std::true_type, Mask>::value))>::type>
unsigned int **count** (Function &&f, const Mask &bitmask=Mask()) const
Count with predicate.
- template<class Reference = bool, class Mask = std::true_type, class = typename std::enable_if<((std::is_convertible<Type, Reference>::value) || (std::is_base_of<Vectorizer, Reference>::value)) && ((std::is_base_of<Vectorizer, Mask>::value) || (std::is_same<std::true_type, Mask>::value))>::type>
bool **all** (const Reference &r=true, const Mask &bitmask=Mask()) const
All values equal.

- template<class Function , class Mask = std::true_type, class = typename std::enable_if<(!std::is_convertible<Type, typename std::decay<Function>::value) && (!std::is_base_of<Vectorizer, typename std::decay<Function>::value)) && ((std::is_base_of<Vectorizer, Mask>::value) || (std::is_same<std::true_type, Mask>::value))>::type>
 bool all (Function &&f, const Mask &bitmask=Mask()) const

All values satisfying the predicate.
- template<class Reference = bool, class Mask = std::true_type, class = typename std::enable_if<((std::is_convertible<Type, Reference>::value) || (std::is_base_of<Vectorizer, Reference>::value)) && ((std::is_base_of<Vectorizer, Mask>::value) || (std::is_same<std::true_type, Mask>::value))>::type>
 bool any (const Reference &r=true, const Mask &bitmask=Mask()) const

Any value equal.
- template<class Function , class Mask = std::true_type, class = typename std::enable_if<((!std::is_convertible<Type, typename std::decay<Function>::value) && (!std::is_base_of<Vectorizer, typename std::decay<Function>::value)) && ((std::is_base_of<Vectorizer, Mask>::value) || (std::is_same<std::true_type, Mask>::value))>::type>
 bool any (Function &&f, const Mask &bitmask=Mask()) const

Any value satisfying the predicate.
- template<class Reference = bool, class Mask = std::true_type, class = typename std::enable_if<((std::is_convertible<Type, Reference>::value) || (std::is_base_of<Vectorizer, Reference>::value)) && ((std::is_base_of<Vectorizer, Mask>::value) || (std::is_same<std::true_type, Mask>::value))>::type>
 bool none (const Reference &r=true, const Mask &bitmask=Mask()) const

No value equal.
- template<class Function , class Mask = std::true_type, class = typename std::enable_if<((!std::is_convertible<Type, typename std::decay<Function>::value) && (!std::is_base_of<Vectorizer, typename std::decay<Function>::value)) && ((std::is_base_of<Vectorizer, Mask>::value) || (std::is_same<std::true_type, Mask>::value))>::type>
 bool none (Function &&f, const Mask &bitmask=Mask()) const

No value satisfying the predicate.

Static Public Member Functions

Static vectorization

- static constexpr unsigned int **size** ()

Get the size of the container.
- static constexpr bool **constant** ()

Get whether the container has a constant size.
- static constexpr bool **boolean** ()

Get whether the container has a boolean type.
- static constexpr std::array<Kind, sizeof...(Parameters)> **parameters** ()

Get the template parameters.
- static Type **type** ()

Get the data type.

Size

- static constexpr bool **empty** ()

Get whether the container is empty.
- static constexpr unsigned int **capacity** ()

Get the capacity of the container.
- static constexpr unsigned int **tbytes** ()

Get the size of the data type.
- static constexpr unsigned long long int **bytes** ()

Get the size in bytes.
- static unsigned long long int **space** ()

Get the maximum available space.

Predefined

- static Crtp< bool, Parameters...> **mask** (const bool value=true)
Default mask creation.
- template<class Container , class = typename std::enable_if<(!std::is_base_of<Vectorizer, Container>::value) && ((std::is_convertible<typename std::remove_reference<decltype(std::declval<Container>()[0])>::type, bool>::value) && ((std::is_void<decltype(std::declval<Container>().flip())>::value) || (std::is_reference<decltype(std::declval<Container>().flip())>::value))>::type>
static Crtp< bool, Parameters...> mask (const Container &container)
Standard boolean container mask creation.
- template<typename OtherType , class... Misc, class = typename std::enable_if<std::is_convertible<OtherType, bool>::value>::type>
static Crtp< bool, Parameters...> mask (const std::initializer_list< OtherType > &source, const Misc &...misc)
Initializer list mask creation.
- template<class... Misc, class = typename std::enable_if<sizeof...(Misc) != 0>::type>
static Crtp< bool, Parameters...> mask (const Misc &...misc)
Generic mask creation.

Test

- static int **example** ()
Example function.

Protected Member Functions

Protected lifecycle

- **~StaticVectorizer** ()
Protected destructor.

Friends

Operators : with lhs value

- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class >
SelfCrtp< typename std::common_type< SelfType, OtherType >::type, SelfParameters...> operator+ (const OtherType &lhs, const **StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...>** &rhs)
Addition with lhs value.
- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class >
SelfCrtp< typename std::common_type< SelfType, OtherType >::type, SelfParameters...> operator- (const OtherType &lhs, const **StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...>** &rhs)
Subtraction with lhs value.
- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class >
SelfCrtp< typename std::common_type< SelfType, OtherType >::type, SelfParameters...> operator* (const OtherType &lhs, const **StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...>** &rhs)
Multiplication with lhs value.

- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class >
`SelfCrtp< typename
std::common_type< SelfType,
OtherType >::type,
SelfParameters...> operator/ (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp,
SelfType, SelfParameters...> &rhs)`
Division with lhs value.
- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class >
`SelfCrtp< typename
std::common_type< SelfType,
OtherType >::type,
SelfParameters...> operator% (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp,
SelfType, SelfParameters...> &rhs)`
Modulo with lhs value.
- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class >
`SelfCrtp< typename
std::common_type< SelfType,
OtherType >::type,
SelfParameters...> operator& (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp,
SelfType, SelfParameters...> &rhs)`
Bitwise AND with lhs value.
- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class >
`SelfCrtp< typename
std::common_type< SelfType,
OtherType >::type,
SelfParameters...> operator| (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp,
SelfType, SelfParameters...> &rhs)`
Bitwise OR with lhs value.
- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class >
`SelfCrtp< typename
std::common_type< SelfType,
OtherType >::type,
SelfParameters...> operator^ (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp,
SelfType, SelfParameters...> &rhs)`
Bitwise XOR with lhs value.
- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class >
`SelfCrtp< OtherType,
SelfParameters...> operator<< (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp,
SelfType, SelfParameters...> &rhs)`
Bitwise left shift with lhs value.
- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class >
`SelfCrtp< OtherType,
SelfParameters...> operator>> (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp,
SelfType, SelfParameters...> &rhs)`
Bitwise right shift with lhs value.
- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class >
`SelfCrtp< bool, SelfParameters...> operator&& (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> &rhs)`
Logical AND with lhs value.
- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class >

`SelfCrtp< bool, SelfParameters...> operator|| (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> &rhs)`

Logical OR with lhs value.

- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class >

`SelfCrtp< bool, SelfParameters...> operator== (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> &rhs)`

Equal to with lhs value.

- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class >

`SelfCrtp< bool, SelfParameters...> operator!= (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> &rhs)`

Not equal to with lhs value.

- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class >

`SelfCrtp< bool, SelfParameters...> operator> (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> &rhs)`

Greater than with lhs value.

- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class >

`SelfCrtp< bool, SelfParameters...> operator< (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> &rhs)`

Less than with lhs value.

- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class >

`SelfCrtp< bool, SelfParameters...> operator>= (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> &rhs)`

Greater than or equal to with lhs value.

- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class >

`SelfCrtp< bool, SelfParameters...> operator<= (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> &rhs)`

Less than or equal to with lhs value.

Stream

- template<typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters>

`std::ostream & operator<< (std::ostream &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> &rhs)`

Output stream operator.

- template<typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters>

`std::istream & operator>> (std::istream &lhs, StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> &rhs)`

Input stream operator.

6.46.1 Detailed Description

`template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type, Kind... Parameters>class magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >`

Helper class for generic constant size vectorization.

Provides vectorization for constant size containers thanks to the curiously recurring template pattern (CRTP) trick. To use it, one has to derive from this class and pass the derived class itself as the CRTP parameter. The derived classes have to satisfy the conditions required by the [Vectorizer](#) base class and have to implement the following functions required by CRTP :

- `operator[](const unsigned int)`

One can also modify members like `operator()` to change the behaviour of the function.

Template Parameters

<i>Kind</i>	Kind of arguments.
<i>Size</i>	Number of elements.
<i>Crtp</i>	Derived CRTP class.
<i>Type</i>	Data type.
<i>Parameters</i>	List of parameters.

6.46.2 Constructor & Destructor Documentation

6.46.2.1 `template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::~StaticVectorizer() [inline], [protected], [default]`

Protected destructor.

Does nothing.

6.46.3 Member Function Documentation

6.46.3.1 `template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<class Reference , class Mask , class > bool magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::all (const Reference & r = true, const Mask & bitmask = Mask ()) const [inline]`

All values equal.

Checks if the comparison with the provided reference returns true for all elements in the specified region. Note that before any comparison, the values in the container are casted to the reference data type. With no argument, this function returns true if the whole contents is non-null (true). It returns true if the container is empty.

Template Parameters

<i>Reference</i>	(Reference type.)
<i>Mask</i>	(Mask type.)

Parameters

<i>in</i>	<i>r</i>	Reference for comparison : value or vectorized container.
<i>in</i>	<i>bitmask</i>	Boolean mask.

Returns

Copy of the boolean result.

6.46.3.2 `template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<class Function , class Mask , class > bool magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::all (Function && f, const Mask & bitmask = Mask ()) const [inline]`

All values satisfying the predicate.

Checks if the unary predicate returns true for all elements in the specified region. It returns true if the container is empty.

Template Parameters

<i>Function</i>	(Function type : <code>bool (Type) .</code>)
<i>Mask</i>	(Mask type.)

Parameters

in	<i>f</i>	Predicate <code>bool (Type) .</code>
in	<i>bitmask</i>	Boolean mask.

Returns

Copy of the boolean result.

6.46.3.3 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind...

Parameters> template<class Reference , class Mask , class > bool magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::any (`const Reference & r = true, const Mask & bitmask = Mask ()`) const [inline]

Any value equal.

Checks if the comparison with the provided reference returns true for any element in the specified region. Note that before any comparison, the values in the container are casted to the reference data type. With no argument, this function returns true if the whole contents is non-null (true). It returns false if the container is empty.

Template Parameters

<i>Reference</i>	(Reference type.)
<i>Mask</i>	(Mask type.)

Parameters

in	<i>r</i>	Reference for comparison : value or vectorized container.
in	<i>bitmask</i>	Boolean mask.

Returns

Copy of the boolean result.

6.46.3.4 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind...

Parameters> template<class Function , class Mask , class > bool magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::any (`Function && f, const Mask & bitmask = Mask ()`) const [inline]

Any value satisfying the predicate.

Checks if the unary predicate returns true for any element in the specified region. It returns false if the container is empty.

Template Parameters

<i>Function</i>	(Function type : <code>bool (Type) .</code>)
<i>Mask</i>	(Mask type.)

Parameters

in	<i>f</i>	Predicate <code>bool (Type).</code>
in	<i>bitmask</i>	Boolean mask.

Returns

Copy of the boolean result.

6.46.3.5 `template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename Return , class Function , class... Args, class > Crtp< Return, Parameters...> magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::apply (Function && f, Args &&... args) const [inline]`

Application of a function object.

Applies a function object to each element of the container and returns a copy of the result. For a result `y`, an element `x`, a function `f` and for extra arguments `args...`, an equivalent expression is : `y = f(x, args...)`.

Template Parameters

<i>Return</i>	Return type.
<i>Function</i>	(Function type : <code>Return (Type, Args...).</code>)
<i>Args</i>	(Extra types.)

Parameters

in	<i>f</i>	Function object <code>Return (Type, Args...).</code>
in	<i>args</i>	Extra arguments of the function.

Returns

Copy.

Warning

The size of the extra arguments is not checked.

6.46.3.6 `template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename Return , class Mask , class Function, class... Args, class > Crtp< Return, Parameters...> magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::apply (const Mask & bitmask, Function && f, Args &&... args) const [inline]`

Masked application of a function object.

Applies a function object to each element of the container where the mask is true and returns a copy of the result. For a result `y`, an element `x`, a function `f` and for extra arguments `args...`, an equivalent expression is : `y = f(x, args...)`.

Template Parameters

<i>Return</i>	Return type.
<i>Mask</i>	(Mask type.)
<i>Function</i>	(Function type : <code>Return (Type, Args...).</code>)
<i>Args</i>	(Extra types.)

Parameters

in	<i>bitmask</i>	Boolean mask.
in	<i>f</i>	Function object <code>Return(Type, Args...)</code> .
in	<i>args</i>	Extra arguments of the function.

Returns

Copy.

Warning

The size of the extra arguments is not checked.

6.46.3.7 `template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type, Kind... Parameters> template<typename OtherType, class... Misc, class > Crtp< Type, Parameters...> & magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::assign (const std::initializer_list< OtherType > & source, const Misc &... misc) [inline]`

Initializer list assignment.

Provides an assignment from an initializer list equivalent to a call to a constructor. Before any operation, the contents is reinitialized to its default value.

Template Parameters

<i>OtherType</i>	(Other data type.)
<i>Misc</i>	(Miscellaneous types.)

Parameters

in	<i>source</i>	Source of the copy.
in	<i>misc</i>	Miscellaneous arguments.

Returns

Self reference.

6.46.3.8 `template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type, Kind... Parameters> template<class... Misc> Crtp< Type, Parameters...> & magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::assign (const Misc &... misc) [inline]`

Generic assignment.

Provides a generic assignment equivalent to a call to a constructor. Before any operation, the contents is reinitialized to its default value.

Template Parameters

<i>Misc</i>	(Miscellaneous types.)
-------------	------------------------

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Self reference.

6.46.3.9 `template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type, Kind... Parameters> Type & magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::at (const unsigned int i) [inline]`

Monodimensional access with range-check.

Provides a monodimensional access to the element with a range-check. Due to the check this function is slower than the [] or the () operator.

Parameters

in	<i>i</i>	Element index.
----	----------	----------------

Returns

Reference to the element.

Exceptions

<code>std::out_of_range</code>	Out of range.
--------------------------------	---------------

6.46.3.10 `template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type, Kind... Parameters> const Type & magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::at (const unsigned int i) const [inline]`

Immutable monodimensional access with range-check.

Provides a monodimensional access to the element with a range-check. Due to the check this function is slower than the [] or the () operator.

Parameters

in	<i>i</i>	Element index.
----	----------	----------------

Returns

Immutable reference to the element.

Exceptions

<code>std::out_of_range</code>	Out of range.
--------------------------------	---------------

6.46.3.11 `template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type, Kind... Parameters> Type & magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::back (const unsigned int i = 0) [inline]`

Monodimensional access to the i-th element from the end.

Returns a reference to the i-th last element in the container without doing any range check.

Parameters

in	i	Element index.
----	---	----------------

Returns

Reference to the element.

6.46.3.12 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> const Type & magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::back (const unsigned int i = 0) const [inline]

Immutable monodimensional access to the i-th element from the end.

Returns a reference to the i-th last element in the container without doing any range check.

Parameters

in	i	Element index.
----	---	----------------

Returns

Immutable reference to the element.

6.46.3.13 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> constexpr bool magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::boolean () [static]

Get whether the container has a boolean type.

Returns true if the container has a boolean type, false otherwise. This function is required by the vectorization mechanism.

Returns

Copy of true if the container has a boolean type.

6.46.3.14 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> constexpr unsigned long long int magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::bytes () [static]

Get the size in bytes.

Returns the total size in bytes which is the number of elements multiplied by the size in bytes of an element. This does not take into account alignment bytes : to get the real memory imprint use the `sizeof()` function.

Returns

Copy of the size in bytes.

6.46.3.15 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> constexpr unsigned int magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::capacity () [static]

Get the capacity of the container.

Returns the capacity of the container, which is equal to its size.

Returns

Copy of the capacity.

6.46.3.16 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename OtherType , class > Crtp< OtherType, Parameters...> magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::cast() const [inline]

Cast to a different data type.

Returns a copy of the container converted to another data type.

Template Parameters

<i>OtherType</i>	Other data type.
------------------	------------------

Returns

Copy.

6.46.3.17 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<class GenericType > Crtp< Type, Parameters...> magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::change(const GenericType & source, const unsigned int pos, const unsigned int num = 1) [inline]

Change an element of the container.

Provides a single element assignment to modify the contents. This function is well suited for chaining assignment. This is equivalent to the [put\(\)](#) function but it operates on a copy and not on the container itself.

Template Parameters

<i>GenericType</i>	(Value or vectorized type.)
--------------------	-----------------------------

Parameters

in	<i>source</i>	Source of the copy.
in	<i>pos</i>	Starting position of the copy.
in	<i>num</i>	Number of elements to copy.

Returns

Self reference.

6.46.3.18 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> Crtp< Type, Parameters...> & magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::clear() [inline]

Clear contents.

Clear the whole contents and sets the size to zero.

Returns

Self reference.

Exceptions

<code>std::length_error</code>	The container cannot be resized.
--------------------------------	----------------------------------

6.46.3.19 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename Result , class Function , class Arg , class... Args, typename Return > Crtp< Return, Parameters...> **magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >**::combine (Function && f, const StaticVectorizer< Kind, Size, Crtp, Arg, Parameters...> & arg, const StaticVectorizer< Kind, Size, Crtp, Args, Parameters...> &... args) const [inline]

Combination by a function object.

Combines several vectorized containers using a function object. It is equivalent to a transversal reduction operation.

Template Parameters

<i>Result</i>	Specified return type.
<i>Function</i>	(Function type : <code>Return (Return, Type)</code> .)
<i>Arg</i>	(First argument type.)
<i>Args</i>	(Other argument types.)
<i>Return</i>	Return type.

Parameters

in	<i>f</i>	Function object <code>Return (Return, Type)</code> .
in	<i>arg</i>	First argument.
in	<i>args</i>	Other arguments.

Returns

Copy.

6.46.3.20 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename Return , class Function > Crtp< Return, Parameters...> **magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >**::combine (Function &&) const [inline]

Unique combination by a function object.

Does nothing.

Template Parameters

<i>Return</i>	Return type.
<i>Function</i>	(Function type : <code>Return (Return, Type)</code> .)

Returns

Copy.

6.46.3.21 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> constexpr bool **magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >**::constant () [static]

Get whether the container has a constant size.

Returns true if the container has a constant size, false otherwise. This function is required by the vectorization mechanism.

Returns

Copy of true.

6.46.3.22 `template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> Crtp< Type, Parameters...> magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::copy () const [inline]`

Copy.

Returns a copy of the container.

Returns

Copy.

6.46.3.23 `template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<class Reference , class Mask , class > unsigned int magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::count (const Reference & r = true, const Mask & bitmask = Mask ()) const [inline]`

Count values.

Count the number of valid comparisons with the provided reference in the specified region. Note that before any comparison, the values in the container are casted to the reference data type. With no argument, this function counts the number of non-null values (true) in the container.

Template Parameters

<i>Reference</i>	(Reference type.)
<i>Mask</i>	(Mask type.)

Parameters

in	<i>r</i>	Reference for comparison : value or vectorized container.
in	<i>bitmask</i>	Boolean mask.

Returns

Copy of the valid counts.

6.46.3.24 `template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<class Function , class Mask , class > unsigned int magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::count (Function && f, const Mask & bitmask = Mask ()) const [inline]`

Count with predicate.

Count the numbers of times the predicate is true in the specified region.

Template Parameters

<i>Function</i>	(Function type : <code>bool (Type) .</code>)
<i>Mask</i>	(Mask type.)

Parameters

in	<i>f</i>	Predicate <code>bool (Type).</code>
in	<i>bitmask</i>	Boolean mask.

Returns

Copy of the valid counts.

6.46.3.25 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> Type & magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::cycle (const int *i*) [inline]

Cyclic monodimensional access to the contents.

Provides a cyclic access to the contents, using the index modulo. Negative indexes are supported. It allows to iterate several times over the contents just by incrementing the provided index.

Parameters

in	<i>i</i>	Element index.
----	----------	----------------

Returns

Reference to the element.

6.46.3.26 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> const Type & magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::cycle (const int *i*) const [inline]

Immutable cyclic monodimensional access to the contents.

Provides a cyclic access to the contents, using the index modulo. Negative indexes are supported. It allows to iterate several times over the contents just by incrementing the provided index.

Parameters

in	<i>i</i>	Element index.
----	----------	----------------

Returns

Immutable reference to the element.

6.46.3.27 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> constexpr bool magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::empty () [static]

Get whether the container is empty.

Returns the result of the comparison between the size and zero.

Returns

Copy of the result of the test.

6.46.3.28 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind...> template<class GenericType > bool magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::eq (const GenericType & rhs) const [inline]

Compare for equality.

Returns true if all elements of the containers are equal, returns false otherwise.

Template Parameters

<i>GenericType</i>	(Value or vectorized type.)
--------------------	-----------------------------

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Copy of the result.

6.46.3.29 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind...> int magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::example () [static]

Example function.

Tests and demonstrates the use of [StaticVectorizer](#).

Returns

0 if no error.

6.46.3.30 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind...> template<typename OtherType , class... Misc, class > Crtp< Type, Parameters...> & magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::fill (const std::initializer_list< OtherType > & source, const Misc &... misc) [inline]

Initializer list fill.

Fills the contents using the [set \(\)](#) function and without reinitializing the contents before any operation. This is equivalent to the [replace \(\)](#) function but it operates on the container itself and not on a copy.

Template Parameters

<i>OtherType</i>	(Other data type.)
<i>Misc</i>	(Miscellaneous types.)

Parameters

in	<i>source</i>	Source of the copy.
in	<i>misc</i>	Miscellaneous arguments.

Returns

Self reference.

6.46.3.31 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<class... Misc> Crtp< Type, Parameters...> & magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::fill (const Misc &... misc) [inline]

Generic fill.

Fills the contents using the `set()` function and without reinitializing the contents before any operation. This is equivalent to the `replace()` function but it operates on the container itself and not on a copy.

Template Parameters

<i>Misc</i>	(Miscellaneous types.)
-------------	------------------------

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Self reference.

6.46.3.32 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> Type & magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::front (const unsigned int *i* = 0) [inline]

Monodimensional access to the *i*-th element from the beginning.

Returns a reference to the *i*-th first element in the container without doing any range check.

Parameters

in	<i>i</i>	Element index.
----	----------	----------------

Returns

Reference to the element.

6.46.3.33 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> const Type & magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::front (const unsigned int *i* = 0) const [inline]

Immutable monodimensional access to the *i*-th element from the beginning.

Returns a reference to the *i*-th first element in the container without doing any range check.

Parameters

in	<i>i</i>	Element index.
----	----------	----------------

Returns

Immutable reference to the element.

6.46.3.34 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> Crtp< bool, Parameters...> **magrathea::StaticVectorizer**< Kind, Size, Crtp, Type, Parameters >::mask (const bool *value* = true) [inline], [static]

Default mask creation.

Creates a mask from a boolean value, which is true per default.

Parameters

in	<i>value</i>	Boolean value used to create the mask.
----	--------------	--

Returns

Boolean copy.

6.46.3.35 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<class Container , class > Crtp< bool, Parameters...> **magrathea::StaticVectorizer**< Kind, Size, Crtp, Type, Parameters >::mask (const Container & *container*) [inline], [static]

Standard boolean container mask creation.

Creates a mask from a standard boolean container, which can be a std::bitset or a std::vector<bool>.

Template Parameters

<i>Container</i>	(Boolean container type.)
------------------	---------------------------

Parameters

in	<i>container</i>	Boolean container.
----	------------------	--------------------

Returns

Boolean copy.

6.46.3.36 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename OtherType , class... Misc, class > Crtp< bool, Parameters...> **magrathea::StaticVectorizer**< Kind, Size, Crtp, Type, Parameters >::mask (const std::initializer_list< OtherType > & *source*, const Misc &... *misc*) [inline], [static]

Initializer list mask creation.

Creates a mask from an initializer list by calling the associated constructor.

Template Parameters

<i>OtherType</i>	(Other data type.)
<i>Misc</i>	(Miscellaneous types.)

Parameters

in	<i>source</i>	Source of the copy.
in	<i>misc</i>	Miscellaneous arguments.

Returns

Boolean copy.

6.46.3.37 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<class... Misc, class > Crtp< bool, Parameters...> magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::mask (const Misc &... misc) [inline], [static]

Generic mask creation.

Creates a mask from generic arguments by calling the associated constructor.

Template Parameters

<i>Misc</i>	(Miscellaneous types.)
-------------	------------------------

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Boolean copy.

6.46.3.38 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<class Mask , class > const Type & magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::max (const Mask & bitmask =Mask()) const [inline]

Maximum element.

Returns a reference to the maximum element of the container or masked container.

Template Parameters

<i>Mask</i>	(Mask type.)
-------------	--------------

Parameters

in	<i>bitmask</i>	Boolean mask.
----	----------------	---------------

Returns

Immutable reference to the element.

Exceptions

<i>std::runtime_error</i>	Empty search.
---------------------------	---------------

6.46.3.39 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<class Mask , class > const Type & magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::min (const Mask & bitmask = Mask ()) const [inline]

Minimum element.

Returns a reference to the minimum element of the container or masked container.

Template Parameters

<i>Mask</i>	(Mask type.)
-------------	--------------

Parameters

<i>in</i>	<i>bitmask</i>	Boolean mask.
-----------	----------------	---------------

Returns

Immutable reference to the element.

Exceptions

<i>std::runtime_error</i>	Empty search.
---------------------------	---------------

6.46.3.40 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename Return , class Function , class... Args, class > Crtp< Type, Parameters...> & magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::modify (Function && f, Args &&... args) [inline]

Modification by a function object.

Modifies the container by applying a function object to each element. For an element *x*, a function *f* and for extra arguments *args...*, an equivalent expression is : *x* = *f* (*x*, *args...*). The return type is used as an internal cast before affection.

Template Parameters

<i>Return</i>	Return type.
<i>Function</i>	(Function type : Type (Type, Args...).)
<i>Args</i>	(Extra types.)

Parameters

<i>in</i>	<i>f</i>	Function object Type (Type, Args...).
<i>in</i>	<i>args</i>	Extra arguments of the function.

Returns

Self reference.

Warning

The size of the extra arguments is not checked.

6.46.3.41 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename Return , class Mask , class Function, class... Args, class > Crtp< Type, Parameters...> & magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::modify (const Mask & bitmask, Function && f, Args &&... args) [inline]

Masked modification by a function object.

Modifies the container by applying a function object to each element where the mask is true. For an element x , a function f and for extra arguments $args\dots$, an equivalent expression is : $x = f(x, args\dots)$. The return type is used as an internal cast before affectation.

Template Parameters

<i>Return</i>	Return type.
<i>Mask</i>	(Mask type.)
<i>Function</i>	(Function type : Type (Type, Args...).)
<i>Args</i>	(Extra types.)

Parameters

in	<i>bitmask</i>	Boolean mask.
in	<i>f</i>	Function object Type (Type, Args...).
in	<i>args</i>	Extra arguments of the function.

Returns

Self reference.

Warning

The size of the extra arguments is not checked.

6.46.3.42 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<class GenericType > bool magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::ne (const GenericType & rhs) const [inline]

Compare for difference.

Returns true if at least one element is different in the two containers, returns false otherwise.

Template Parameters

<i>GenericType</i>	(Value or vectorized type.)
--------------------	-----------------------------

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Copy of the result.

6.46.3.43 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<class Reference , class Mask , class > bool magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::none (const Reference & r = true, const Mask & bitmask = Mask ()) const [inline]

No value equal.

Checks if the comparison with the provided reference returns true for no element in the specified region. Note that before any comparison, the values in the container are casted to the reference data type. With no argument, this function returns true if the whole contents is null (false). It returns true if the container is empty.

Template Parameters

<i>Reference</i>	(Reference type.)
<i>Mask</i>	(Mask type.)

Parameters

in	<i>r</i>	Reference for comparison : value or vectorized container.
in	<i>bitmask</i>	Boolean mask.

Returns

Copy of the boolean result.

6.46.3.44 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<class Function , class Mask , class > bool magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::none (Function && f, const Mask & bitmask = Mask ()) const [inline]

No value satisfying the predicate.

Checks if the unary predicate returns true for no element in the specified region. It returns true if the container is empty.

Template Parameters

<i>Function</i>	(Function type : bool (Type) .)
<i>Mask</i>	(Mask type.)

Parameters

in	<i>f</i>	Predicate bool (Type) .
in	<i>bitmask</i>	Boolean mask.

Returns

Copy of the boolean result.

6.46.3.45 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> bool magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::null () const [inline]

Check whether all elements are null.

Returns true if all elements are set to their default value, returns false otherwise.

Returns

Copy of the result of the test.

6.46.3.46 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<class... Location> Crtp< Type, Parameters...> & magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::nullify (const Location &... location) [inline]

Set elements to their default values.

Sets the specified elements to their default value using the [set \(\)](#) function for the provided location.

Template Parameters

<i>Location</i>	(Mask or position specifiers.)
-----------------	--------------------------------

Parameters

<i>in</i>	<i>location</i>	Boolean mask or starting position and number of elements.
-----------	-----------------	---

Returns

Self reference.

6.46.3.47 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> Crtp< bool, Parameters...> magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator! () const [inline]

Logical NOT.

Applies the logical NOT operator to each element.

Returns

Boolean copy.

6.46.3.48 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename OtherType , class > Crtp< bool, Parameters...> magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator!= (const StaticVectorizer< Kind, Size, Crtp, OtherType, Parameters...> & rhs) const [inline]

Not equal to.

Applies the not equal to operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
------------------	--------------------

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Boolean copy.

```
6.46.3.49 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename OtherType , class > Crtp< bool, Parameters...> magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator!= ( const OtherType & rhs ) const [inline]
```

Not equal to with rhs value.

Applies the not equal to operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
------------------	--------------------

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Boolean copy.

```
6.46.3.50 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename OtherType , class > Crtp< typename std::common_type< Type, OtherType >::type, Parameters...> magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator% ( const StaticVectorizer< Kind, Size, Crtp, OtherType, Parameters...> & rhs ) const [inline]
```

Modulo.

Applies the modulo operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
------------------	--------------------

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Common type copy.

6.46.3.51 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename OtherType , class > Crtp< typename std::common_type< Type, OtherType >::type, Parameters...> **magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator% (const OtherType & rhs) const [inline]**

Modulo with rhs value.

Applies the modulo operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
------------------	--------------------

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Common type copy.

6.46.3.52 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<class GenericType > Crtp< Type, Parameters...> & **magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator%= (const GenericType & rhs) [inline]**

Modulo assignment.

Applies the modulo assignment operator to each element.

Template Parameters

<i>GenericType</i>	(Value or vectorized type.)
--------------------	-----------------------------

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Self reference

6.46.3.53 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename OtherType , class > Crtp< typename std::common_type< Type, OtherType >::type, Parameters...> **magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator& (const StaticVectorizer< Kind, Size, Crtp, OtherType, Parameters...> & rhs) const [inline]**

Bitwise AND.

Applies the bitwise AND operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
------------------	--------------------

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Common type copy.

```
6.46.3.54 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind...
Parameters> template<typename OtherType , class > Crtp< typename std::common_type< Type, OtherType
>::type, Parameters...> magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator& (
const OtherType & rhs ) const [inline]
```

Bitwise AND with rhs value.

Applies the bitwise AND operator to each element.

Template Parameters

OtherType	(Other data type.)
-----------	--------------------

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Common type copy.

```
6.46.3.55 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename
Type , Kind... Parameters> template<typename OtherType , class > Crtp< bool, Parameters...>
magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator&& ( const StaticVectorizer<
Kind, Size, Crtp, OtherType, Parameters...> & rhs ) const [inline]
```

Logical AND.

Applies the logical AND operator to each element.

Template Parameters

OtherType	(Other data type.)
-----------	--------------------

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Boolean copy.

```
6.46.3.56 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename
Type , Kind... Parameters> template<typename OtherType , class > Crtp< bool, Parameters...>
magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator&& ( const OtherType & rhs )
const [inline]
```

Logical AND with rhs value.

Applies the logical AND operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
------------------	--------------------

Parameters

<i>in</i>	<i>rhs</i>	Right-hand side.
-----------	------------	------------------

Returns

Boolean copy.

6.46.3.57 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<class GenericType > Crtp< Type, Parameters...> & magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator&= (const GenericType & *rhs*) [inline]

Bitwise AND assignment.

Applies the bitwise AND assignment operator to each element.

Template Parameters

<i>GenericType</i>	(Value or vectorized type.)
--------------------	-----------------------------

Parameters

<i>in</i>	<i>rhs</i>	Right-hand side.
-----------	------------	------------------

Returns

Self reference

6.46.3.58 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> StaticVectorizer< Kind, Size, Crtp, Type, Parameters...> & magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator() () [inline]

Abstract class access.

Casts to the abstract class.

Returns

Self reference.

6.46.3.59 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> const StaticVectorizer< Kind, Size, Crtp, Type, Parameters...> & magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator() () const [inline]

Immutable abstract class access.

Casts to the abstract class.

Returns

Immutable self reference.

6.46.3.60 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> Type & magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator() (const unsigned int *i*) [inline]

Monodimensional access operator.

Provides a monodimensional access to the element. For a monodimensional array, it is equivalent to the [] operator.

Parameters

in	<i>i</i>	Element index.
----	----------	----------------

Returns

Reference to the element.

6.46.3.61 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> const Type & magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator() (const unsigned int *i*) const [inline]

Immutable monodimensional access operator.

Provides a monodimensional access to the element. For a monodimensional array, it is equivalent to the [] operator.

Parameters

in	<i>i</i>	Element index.
----	----------	----------------

Returns

Immutable reference to the element.

6.46.3.62 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename OtherType , class > Crtp< typename std::common_type< Type, OtherType >::type, Parameters...> magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator* (const StaticVectorizer< Kind, Size, Crtp, OtherType, Parameters...> & *rhs*) const [inline]

Multiplication.

Applies the multiplication operator to each element.

Template Parameters

OtherType	(Other data type.)
-----------	--------------------

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Common type copy.

6.46.3.63 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename OtherType , class > Crtp< typename std::common_type< Type, OtherType >::type, Parameters...> **magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >**::operator* (const OtherType & *rhs*) const [inline]

Multiplication with rhs value.

Applies the multiplication operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
------------------	--------------------

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Common type copy.

6.46.3.64 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<class GenericType > Crtp< Type, Parameters...> & **magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >**::operator*=(const GenericType & *rhs*) [inline]

Multiplication assignment.

Applies the multiplication assignment operator to each element.

Template Parameters

<i>GenericType</i>	(Value or vectorized type.)
--------------------	-----------------------------

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Self reference

6.46.3.65 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename OtherType , class > Crtp< typename std::common_type< Type, OtherType >::type, Parameters...> **magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >**::operator+ (const StaticVectorizer< Kind, Size, Crtp, OtherType, Parameters...> & *rhs*) const [inline]

Addition.

Applies the addition operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
------------------	--------------------

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Common type copy.

6.46.3.66 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename OtherType , class > Crtp< typename std::common_type< Type, OtherType >::type, Parameters...> magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator+ (const OtherType & *rhs*) const [inline]

Addition with rhs value.

Applies the addition operator to each element.

Template Parameters

	<i>OtherType</i>	(Other data type.)
--	------------------	--------------------

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Common type copy.

6.46.3.67 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> Crtp< Type, Parameters...> magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator+ () const [inline]

Integer promotion.

Applies the integer promotion operator to each element.

Returns

Copy.

6.46.3.68 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> Crtp< Type, Parameters...> & magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator++ () [inline]

Increment prefix.

Applies the increment prefix operator to each element.

Returns

Self reference.

6.46.3.69 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> Crtp< Type, Parameters...> **magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator++(int)** [inline]

Increment suffix.

Applies the increment suffix operator to each element.

Returns

Copy.

6.46.3.70 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<class GenericType > Crtp< Type, Parameters...> & **magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator+=(const GenericType & rhs)** [inline]

Addition assignment.

Applies the addition assignment operator to each element.

Template Parameters

<i>GenericType</i>	(Value or vectorized type.)
--------------------	-----------------------------

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Self reference

6.46.3.71 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename OtherType , class > Crtp< typename std::common_type< Type, OtherType >::type, Parameters...> **magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator-(const StaticVectorizer< Kind, Size, Crtp, OtherType, Parameters...> & rhs) const** [inline]

Subtraction.

Applies the subtraction operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
------------------	--------------------

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Common type copy.

6.46.3.72 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename OtherType , class > Crtp< typename std::common_type< Type, OtherType >::type, Parameters...> **magrathea::StaticVectorizer**< Kind, Size, Crtp, Type, Parameters >::operator- (const OtherType & rhs) const [inline]

Subtraction with rhs value.

Applies the subtraction operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
------------------	--------------------

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Common type copy.

6.46.3.73 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> Crtp< Type, Parameters...> **magrathea::StaticVectorizer**< Kind, Size, Crtp, Type, Parameters >::operator- () const [inline]

Additive inverse.

Applies the additive inverse operator to each element.

Returns

Copy.

6.46.3.74 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> Crtp< Type, Parameters...> & **magrathea::StaticVectorizer**< Kind, Size, Crtp, Type, Parameters >::operator-- () [inline]

Decrement prefix.

Applies the decrement prefix operator to each element.

Returns

Self reference.

6.46.3.75 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> Crtp< Type, Parameters...> **magrathea::StaticVectorizer**< Kind, Size, Crtp, Type, Parameters >::operator-- (int) [inline]

Decrement suffix.

Applies the decrement suffix operator to each element.

Returns

Copy.

6.46.3.76 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<class GenericType > Crtp< Type, Parameters...> & magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator=(const GenericType & rhs) [inline]

Subtraction assignment.

Applies the subtraction assignment operator to each element.

Template Parameters

<i>GenericType</i>	(Value or vectorized type.)
--------------------	-----------------------------

Parameters

<i>in</i>	<i>rhs</i>	Right-hand side.
-----------	------------	------------------

Returns

Self reference

6.46.3.77 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename OtherType , class > Crtp< typename std::common_type< Type, OtherType >::type, Parameters...> magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator/ (const StaticVectorizer< Kind, Size, Crtp, OtherType, Parameters...> & rhs) const [inline]

Division.

Applies the division operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
------------------	--------------------

Parameters

<i>in</i>	<i>rhs</i>	Right-hand side.
-----------	------------	------------------

Returns

Common type copy.

6.46.3.78 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename OtherType , class > Crtp< typename std::common_type< Type, OtherType >::type, Parameters...> magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator/ (const OtherType & rhs) const [inline]

Division with rhs value.

Applies the division operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
------------------	--------------------

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Common type copy.

6.46.3.79 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<class GenericType > Crtp< Type, Parameters...> & magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator/= (const GenericType & *rhs*) [inline]

Division assignment.

Applies the division assignment operator to each element.

Template Parameters

<i>GenericType</i>	(Value or vectorized type.)
--------------------	-----------------------------

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Self reference

6.46.3.80 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename OtherType , class > Crtp< bool, Parameters...> magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator<< (const StaticVectorizer< Kind, Size, Crtp, OtherType, Parameters...> & *rhs*) const [inline]

Less than.

Applies the less than operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
------------------	--------------------

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Boolean copy.

6.46.3.81 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename OtherType , class > Crtp< bool, Parameters...> magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator<< (const OtherType & rhs) const [inline]

Less than with rhs value.

Applies the less than operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
------------------	--------------------

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Boolean copy.

6.46.3.82 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename OtherType , class > Crtp< Type, Parameters...> magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator<<< (const StaticVectorizer< Kind, Size, Crtp, OtherType, Parameters...> & rhs) const [inline]

Bitwise left shift.

Applies the bitwise left shift operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
------------------	--------------------

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Copy.

6.46.3.83 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename OtherType , class > Crtp< Type, Parameters...> magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator<<< (const OtherType & rhs) const [inline]

Bitwise left shift with rhs value.

Applies the bitwise left shift operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
------------------	--------------------

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Copy.

6.46.3.84 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<class GenericType > Crtp< Type, Parameters...> & magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator<<= (const GenericType & rhs) [inline]

Bitwise left shift assignment.

Applies the bitwise left shift assignment operator to each element.

Template Parameters

<i>GenericType</i>	(Value or vectorized type.)
--------------------	-----------------------------

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Self reference

6.46.3.85 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename OtherType , class > Crtp< bool, Parameters...> magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator<= (const StaticVectorizer< Kind, Size, Crtp, OtherType, Parameters...> & rhs) const [inline]

Less than or equal to.

Applies the less than or equal to operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
------------------	--------------------

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Boolean copy.

6.46.3.86 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename OtherType , class > Crtp< bool, Parameters...> magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator<= (const OtherType & rhs) const [inline]

Less than or equal to with rhs value.

Applies the less than or equal to operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
------------------	--------------------

Parameters

<i>in</i>	<i>rhs</i>	Right-hand side.
-----------	------------	------------------

Returns

Boolean copy.

6.46.3.87 template<typename Kind, unsigned int Size, template< typename, Kind... > class Crtp, typename Type, Kind... Parameters> Crtp< Type, Parameters... > & magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator=(const StaticVectorizer< Kind, Size, Crtp, Type, Parameters... > & rhs) [inline]

Copy assignment operator.

Copies the contents of the container.

Parameters

<i>in</i>	<i>rhs</i>	Right-hand side.
-----------	------------	------------------

Returns

Self reference.

6.46.3.88 template<typename Kind , unsigned int Size, template< typename, Kind... > class Crtp, typename Type , Kind... Parameters> template<typename OtherType , class > Crtp< Type, Parameters... > & magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator=(const std::initializer_list< OtherType > & rhs) [inline]

Initializer list assignment operator.

Provides an initializer list assignment. The assignment is delegated to the [set\(\)](#) helper function thanks to the following call : `set(*this, rhs)`.

Template Parameters

<i>OtherType</i>	(Other data type.)
------------------	--------------------

Parameters

<i>in</i>	<i>rhs</i>	Right-hand side.
-----------	------------	------------------

Returns

Self reference.

6.46.3.89 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<class Misc > Crtp< Type, Parameters...> & magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator=(const Misc & rhs) [inline]

Copy assignment operator.

Provides a generic assignment. The assignment is delegated to the `set()` helper function. Conversion assignment, value assignment and assignment are provided through this function thanks to the following call : `set(*this, misc...)`.

Template Parameters

<i>Misc</i>	(Miscellaneous type.)
-------------	-----------------------

Parameters

<i>in</i>	<i>rhs</i>	Right-hand side.
-----------	------------	------------------

Returns

Self reference.

6.46.3.90 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename OtherType , class > Crtp< bool, Parameters...> magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator==(const StaticVectorizer< Kind, Size, Crtp, OtherType, Parameters...> & rhs) const [inline]

Equal to.

Applies the equal to operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
------------------	--------------------

Parameters

<i>in</i>	<i>rhs</i>	Right-hand side.
-----------	------------	------------------

Returns

Boolean copy.

6.46.3.91 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename OtherType , class > Crtp< bool, Parameters...> magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator==(const OtherType & rhs) const [inline]

Equal to with rhs value.

Applies the equal to operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
------------------	--------------------

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Boolean copy.

6.46.3.92 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename OtherType , class > Crtp< bool, Parameters...> magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator> (const StaticVectorizer< Kind, Size, Crtp, OtherType, Parameters...> & rhs) const [inline]

Greater than.

Applies the greater than operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
------------------	--------------------

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Boolean copy.

6.46.3.93 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename OtherType , class > Crtp< bool, Parameters...> magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator> (const OtherType & rhs) const [inline]

Greater than with rhs value.

Applies the greater than operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
------------------	--------------------

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Boolean copy.

6.46.3.94 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename OtherType , class > Crtp< bool, Parameters...> magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator>= (const StaticVectorizer< Kind, Size, Crtp, OtherType, Parameters...> & rhs) const [inline]

Greater than or equal to.

Applies the greater than or equal to operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
------------------	--------------------

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Boolean copy.

6.46.3.95 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename OtherType , class > Crtp< bool, Parameters...> magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator>= (const OtherType & rhs) const [inline]

Greater than or equal to with rhs value.

Applies the greater than or equal to operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
------------------	--------------------

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Boolean copy.

6.46.3.96 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename OtherType , class > Crtp< Type, Parameters...> magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator>> (const StaticVectorizer< Kind, Size, Crtp, OtherType, Parameters...> & rhs) const [inline]

Bitwise right shift.

Applies the bitwise right shift operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
------------------	--------------------

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Copy.

```
6.46.3.97 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename OtherType , class > Crtp< Type, Parameters...> magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator>> ( const OtherType & rhs ) const [inline]
```

Bitwise right shift with rhs value.

Applies the bitwise right shift operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
------------------	--------------------

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Copy.

```
6.46.3.98 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<class GenericType > Crtp< Type, Parameters...> & magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator>>= ( const GenericType & rhs ) [inline]
```

Bitwise right shift assignment.

Applies the bitwise right shift assignment operator to each element.

Template Parameters

<i>GenericType</i>	(Value or vectorized type.)
--------------------	-----------------------------

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Self reference

```
6.46.3.99 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> Type & magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator[] ( const unsigned int i ) [inline]
```

Direct access to the element.

Provides a direct access to the specified element. This function is required by the vectorization mechanism.

Parameters

in	i	Index of the element.
----	---	-----------------------

Returns

Reference to the element.

6.46.3.100 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> const Type & magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator[] (const unsigned int i) const [inline]

Immutable direct access to the element.

Provides a constant direct access to the specified element. This function is required by the vectorization mechanism.

Parameters

in	i	Index of the element.
----	---	-----------------------

Returns

Immutable reference to the element.

6.46.3.101 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename OtherType , class > Crtp< typename std::common_type< Type, OtherType >::type, Parameters...> magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator^ (const StaticVectorizer< Kind, Size, Crtp, OtherType, Parameters...> & rhs) const [inline]

Bitwise XOR.

Applies the bitwise XOR operator to each element.

Template Parameters

OtherType	(Other data type.)
-----------	--------------------

Parameters

in	rhs	Right-hand side.
----	-----	------------------

Returns

Common type copy.

6.46.3.102 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename OtherType , class > Crtp< typename std::common_type< Type, OtherType >::type, Parameters...> magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator^ (const OtherType & rhs) const [inline]

Bitwise XOR with rhs value.

Applies the bitwise XOR operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
------------------	--------------------

Parameters

<i>in</i>	<i>rhs</i>	Right-hand side.
-----------	------------	------------------

Returns

Common type copy.

6.46.3.103 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<class GenericType > Crtp< Type, Parameters...> & magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator^=(const GenericType & *rhs*) [inline]

Bitwise XOR assignment.

Applies the bitwise XOR assignment operator to each element.

Template Parameters

<i>GenericType</i>	(Value or vectorized type.)
--------------------	-----------------------------

Parameters

<i>in</i>	<i>rhs</i>	Right-hand side.
-----------	------------	------------------

Returns

Self reference

6.46.3.104 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename OtherType , class > Crtp< typename std::common_type< Type, OtherType >::type, Parameters...> magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator|(const StaticVectorizer< Kind, Size, Crtp, OtherType, Parameters...> & *rhs*) const [inline]

Bitwise OR.

Applies the bitwise OR operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
------------------	--------------------

Parameters

<i>in</i>	<i>rhs</i>	Right-hand side.
-----------	------------	------------------

Returns

Common type copy.

6.46.3.105 template<typename Kind, unsigned int Size, template< typename, Kind... > class Crtp, typename Type , Kind... Parameters> template<typename OtherType , class > Crtp< typename std::common_type< Type, OtherType >::type, Parameters...> magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator|(const OtherType & rhs) const [inline]

Bitwise OR with rhs value.

Applies the bitwise OR operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
------------------	--------------------

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Common type copy.

6.46.3.106 template<typename Kind , unsigned int Size, template< typename, Kind... > class Crtp, typename Type , Kind... Parameters> template<class GenericType > Crtp< Type, Parameters...> & magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator|=(const GenericType & rhs) [inline]

Bitwise OR assignment.

Applies the bitwise OR assignment operator to each element.

Template Parameters

<i>GenericType</i>	(Value or vectorized type.)
--------------------	-----------------------------

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Self reference

6.46.3.107 template<typename Kind, unsigned int Size, template< typename, Kind... > class Crtp, typename Type , Kind... Parameters> template<typename OtherType , class > Crtp< bool, Parameters...> magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator||((const StaticVectorizer< Kind, Size, Crtp, OtherType, Parameters...> & rhs) const [inline]

Logical OR.

Applies the logical OR operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
------------------	--------------------

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Boolean copy.

```
6.46.3.108 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename OtherType , class > Crtp< bool, Parameters...> magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator|| ( const OtherType & rhs )
const [inline]
```

Logical OR with rhs value.

Applies the logical OR operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
------------------	--------------------

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Boolean copy.

```
6.46.3.109 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> Crtp< Type, Parameters...> magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::operator~ ( ) const [inline]
```

Bitwise NOT.

Applies the bitwise NOT operator to each element.

Returns

Copy.

```
6.46.3.110 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> constexpr std::array< Kind, sizeof...(Parameters)> magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::parameters ( ) [static]
```

Get the template parameters.

Returns an array containing the template parameters.

Returns

Copy of an array of parameters.

6.46.3.111 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<class GenericType > Crtp< Type, Parameters...> & magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::put (const GenericType & source, const unsigned int pos, const unsigned int num = 1) [inline]

Put an element in the container.

Provides a single element assignment to modify the contents. This function is well suited for chaining assignment. This is equivalent to the [change \(\)](#) function but it operates on the container itself and not on a copy.

Template Parameters

<i>GenericType</i>	(Value or vectorized type.)
--------------------	-----------------------------

Parameters

in	<i>source</i>	Source of the copy.
in	<i>pos</i>	Starting position of the copy.
in	<i>num</i>	Number of elements to copy.

Returns

Self reference.

6.46.3.112 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename Return , class Function , class > Return magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::reduce (Function && f=Function(), const Return & init=Return()) const [inline]

Reduction by a function object.

Reduces the contents using a binary function object initialized to the *init* value. For each reduced element *x*, the equivalent expression is *result* = *f*(*result*, *x*).

Template Parameters

<i>Return</i>	Return type.
<i>Function</i>	(Function type : <i>Return</i> (<i>Return</i> , <i>Type</i>).)

Parameters

in	<i>f</i>	Function object <i>Return</i> (<i>Return</i> , <i>Type</i>).
in	<i>init</i>	Initial value for the reduction.

Returns

Copy of the result.

6.46.3.113 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename Return , class Mask , class Function , class > Return magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::reduce (const Mask & bitmask, Function && f=Function(), const Return & init=Return()) const [inline]

Reduction by a function object.

Reduces the contents using a binary function object initialized to the *init* value where the mask is true. For each reduced element *x*, the equivalent expression is *result* = *f*(*result*, *x*).

Template Parameters

<i>Return</i>	Return type.
<i>Mask</i>	(Mask type.)
<i>Function</i>	(Function type : Return (Return, Type).)

Parameters

in	<i>bitmask</i>	Boolean mask.
in	<i>f</i>	Function object Return (Return, Type).
in	<i>init</i>	Initial value for the reduction.

Returns

Copy of the result.

6.46.3.114 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<typename OtherType , class... Misc, class > Crtp< Type, Parameters...> magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::replace (const std::initializer_list< OtherType > & source, const Misc &... misc) [inline]

Initializer list replace.

Replaces the contents using the [set \(\)](#) function and without reinitializing the contents before any operation. This is equivalent to the [fill \(\)](#) function but it operates on a copy and not on the container itself.

Template Parameters

<i>OtherType</i>	(Other data type.)
<i>Misc</i>	(Miscellaneous types.)

Parameters

in	<i>source</i>	Source of the copy.
in	<i>misc</i>	Miscellaneous arguments.

Returns

Self reference.

6.46.3.115 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> template<class... Misc> Crtp< Type, Parameters...> magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::replace (const Misc &... misc) [inline]

Generic replace.

Replaces the contents using the [set \(\)](#) function and without reinitializing the contents before any operation. This is equivalent to the [fill \(\)](#) function but it operates on a copy and not on the container itself.

Template Parameters

<i>Misc</i>	(Miscellaneous types.)
-------------	------------------------

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

Returns

Self reference.

6.46.3.116 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> Crtp< Type, Parameters...> & magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::reserve (const unsigned int *n*) [inline]

Reserve new space for the container.

Reserves new space for the container in order to optimize future resize calls.

Parameters

in	<i>n</i>	New size for reservation.
----	----------	---------------------------

Returns

Self reference.

Exceptions

<i>std::length_error</i>	The container cannot be resized.
--------------------------	----------------------------------

6.46.3.117 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> Crtp< Type, Parameters...> & magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::resize (const unsigned int *n*) [inline]

Resize the container.

Resizes the container and returns a reference to it. This function is required by the vectorization mechanism.

Parameters

in	<i>n</i>	New size.
----	----------	-----------

Returns

Self reference.

Exceptions

<i>std::length_error</i>	The container cannot be resized.
--------------------------	----------------------------------

6.46.3.118 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> constexpr unsigned int magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::size () [static]

Get the size of the container.

Returns the current number of elements. This function is required by the vectorization mechanism.

Returns

Copy of the size.

6.46.3.119 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> unsigned long long int magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::space() [inline], [static]

Get the maximum available space.

Returns the `max_size()` of a `std::vector` of the same type.

Returns

Copy of the maximum size.

6.46.3.120 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type, Kind... Parameters> template<class... Location> Crtp< Type, Parameters...> & magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::swap(StaticVectorizer< Kind, Size, Crtp, Type, Parameters...> & rhs, const Location &... location) [inline]

Swap elements by copy.

Swaps the elements of the two containers by copy at the provided location and returns a reference to the left-hand side. The resulting operation is not optimal because it implies a temporary copy.

Template Parameters

<i>Location</i>	(Mask or position specifiers.)
-----------------	--------------------------------

Parameters

in	<i>rhs</i>	Right-hand side.
in	<i>location</i>	Boolean mask or starting position and number of elements.

Returns

Self reference.

6.46.3.121 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> constexpr unsigned int magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::tbytes() [static]

Get the size of the data type.

Returns the size of the element type in bytes.

Returns

Copy of the size of the data type.

6.46.3.122 template<typename Kind , unsigned int Size, template< typename, Kind...> class Crtp, typename Type , Kind... Parameters> Type magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >::type() [inline], [static]

Get the data type.

Returns a copy of the default value of the data type.

Returns

Copy of the default value of the data type.

6.46.4 Friends And Related Function Documentation

6.46.4.1 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type, Kind... Parameters> template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class > SelfCrtp<bool, SelfParameters...> operator!= (const OtherType & lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> & rhs) [friend]

Not equal to with lhs value.

Applies the not equal to operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
<i>SelfKind</i>	(Kind of arguments.)
<i>SelfSize</i>	(Number of elements.)
<i>SelfCrtp</i>	(Derived CRTP class.)
<i>SelfType</i>	(Data type.)
<i>SelfParameters</i>	(List of parameters.)

Parameters

in	<i>lhs</i>	Left-hand side.
in	<i>rhs</i>	Right-hand side.

Returns

Boolean copy.

6.46.4.2 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type, Kind... Parameters> template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class > SelfCrtp<typename std::common_type<SelfType, OtherType>::type, SelfParameters...> operator% (const OtherType & lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> & rhs) [friend]

Modulo with lhs value.

Applies the modulo operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
<i>SelfKind</i>	(Kind of arguments.)
<i>SelfSize</i>	(Number of elements.)
<i>SelfCrtp</i>	(Derived CRTP class.)
<i>SelfType</i>	(Data type.)
<i>SelfParameters</i>	(List of parameters.)

Parameters

in	<i>lhs</i>	Left-hand side.
in	<i>rhs</i>	Right-hand side.

Returns

Common type copy.

6.46.4.3 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type, Kind... Parameters> template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class > SelfCrtp<typename std::common_type<SelfType, OtherType>::type, SelfParameters...> operator& (const OtherType & *lhs*, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> & *rhs*) [friend]

Bitwise AND with *lhs* value.

Applies the bitwise AND operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
<i>SelfKind</i>	(Kind of arguments.)
<i>SelfSize</i>	(Number of elements.)
<i>SelfCrtp</i>	(Derived CRTP class.)
<i>SelfType</i>	(Data type.)
<i>SelfParameters</i>	(List of parameters.)

Parameters

in	<i>lhs</i>	Left-hand side.
in	<i>rhs</i>	Right-hand side.

Returns

Common type copy.

6.46.4.4 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type, Kind... Parameters> template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class > SelfCrtp<bool, SelfParameters...> operator&& (const OtherType & *lhs*, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> & *rhs*) [friend]

Logical AND with *lhs* value.

Applies the logical AND operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
<i>SelfKind</i>	(Kind of arguments.)
<i>SelfSize</i>	(Number of elements.)
<i>SelfCrtp</i>	(Derived CRTP class.)
<i>SelfType</i>	(Data type.)
<i>SelfParameters</i>	(List of parameters.)

Parameters

in	<i>lhs</i>	Left-hand side.
in	<i>rhs</i>	Right-hand side.

Returns

Boolean copy.

6.46.4.5 template<typename Kind, unsigned int Size, template< typename, Kind... > class Crtp, typename Type, Kind... Parameters> template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind... > class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class > SelfCrtp<typename std::common_type<SelfType, OtherType>::type, SelfParameters... > operator*(const OtherType & *lhs*, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters... > & *rhs*) [friend]

Multiplication with *lhs* value.

Applies the multiplication operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
<i>SelfKind</i>	(Kind of arguments.)
<i>SelfSize</i>	(Number of elements.)
<i>SelfCrtp</i>	(Derived CRTP class.)
<i>SelfType</i>	(Data type.)
<i>SelfParameters</i>	(List of parameters.)

Parameters

in	<i>lhs</i>	Left-hand side.
in	<i>rhs</i>	Right-hand side.

Returns

Common type copy.

6.46.4.6 template<typename Kind, unsigned int Size, template< typename, Kind... > class Crtp, typename Type, Kind... Parameters> template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind... > class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class > SelfCrtp<typename std::common_type<SelfType, OtherType>::type, SelfParameters... > operator+(const OtherType & *lhs*, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters... > & *rhs*) [friend]

Addition with *lhs* value.

Applies the addition operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
<i>SelfKind</i>	(Kind of arguments.)
<i>SelfSize</i>	(Number of elements.)
<i>SelfCrtp</i>	(Derived CRTP class.)
<i>SelfType</i>	(Data type.)
<i>SelfParameters</i>	(List of parameters.)

Parameters

in	<i>lhs</i>	Left-hand side.
in	<i>rhs</i>	Right-hand side.

Returns

Common type copy.

6.46.4.7 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type, Kind... Parameters> template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class > SelfCrtp<typename std::common_type<SelfType, OtherType>::type, SelfParameters...> operator- (const OtherType & *lhs*, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> & *rhs*) [friend]

Subtraction with *lhs* value.

Applies the subtraction operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
<i>SelfKind</i>	(Kind of arguments.)
<i>SelfSize</i>	(Number of elements.)
<i>SelfCrtp</i>	(Derived CRTP class.)
<i>SelfType</i>	(Data type.)
<i>SelfParameters</i>	(List of parameters.)

Parameters

in	<i>lhs</i>	Left-hand side.
in	<i>rhs</i>	Right-hand side.

Returns

Common type copy.

6.46.4.8 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type, Kind... Parameters> template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class > SelfCrtp<typename std::common_type<SelfType, OtherType>::type, SelfParameters...> operator/ (const OtherType & *lhs*, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> & *rhs*) [friend]

Division with *lhs* value.

Applies the division operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
<i>SelfKind</i>	(Kind of arguments.)
<i>SelfSize</i>	(Number of elements.)
<i>SelfCrtp</i>	(Derived CRTP class.)
<i>SelfType</i>	(Data type.)
<i>SelfParameters</i>	(List of parameters.)

Parameters

in	<i>lhs</i>	Left-hand side.
in	<i>rhs</i>	Right-hand side.

Returns

Common type copy.

6.46.4.9 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type, Kind... Parameters> template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class > SelfCrtp<bool, SelfParameters...> operator< (const OtherType & *lhs*, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> & *rhs*) [friend]

Less than with *lhs* value.

Applies the less than operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
<i>SelfKind</i>	(Kind of arguments.)
<i>SelfSize</i>	(Number of elements.)
<i>SelfCrtp</i>	(Derived CRTP class.)
<i>SelfType</i>	(Data type.)
<i>SelfParameters</i>	(List of parameters.)

Parameters

in	<i>lhs</i>	Left-hand side.
in	<i>rhs</i>	Right-hand side.

Returns

Boolean copy.

6.46.4.10 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type, Kind... Parameters> template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class > SelfCrtp<OtherType, SelfParameters...> operator<<(const OtherType & *lhs*, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> & *rhs*) [friend]

Bitwise left shift with *lhs* value.

Applies the bitwise left shift operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
<i>SelfKind</i>	(Kind of arguments.)
<i>SelfSize</i>	(Number of elements.)
<i>SelfCrtp</i>	(Derived CRTP class.)
<i>SelfType</i>	(Data type.)
<i>SelfParameters</i>	(List of parameters.)

Parameters

in	<i>lhs</i>	Left-hand side.
in	<i>rhs</i>	Right-hand side.

Returns

Copy.

6.46.4.11 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type, Kind... Parameters> template<typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters> std::ostream& operator<< (std::ostream & *lhs*, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> & *rhs*) [friend]

[Output](#) stream operator.

Adds each element to the stream using the [fill\(\)](#) character as a separator.

Template Parameters

<i>SelfKind</i>	(Kind of arguments.)
<i>SelfSize</i>	(Number of elements.)
<i>SelfCrtp</i>	(Derived CRTP class.)
<i>SelfType</i>	(Data type.)
<i>SelfParameters</i>	(List of parameters.)

Parameters

in,out	<i>lhs</i>	Left-hand side stream.
in	<i>rhs</i>	Right-hand side container.

Returns

[Output](#) stream.

6.46.4.12 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type, Kind... Parameters> template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class > SelfCrtp<bool, SelfParameters...> operator<=(const OtherType & *lhs*, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> & *rhs*) [friend]

Less than or equal to with *lhs* value.

Applies the less than or equal to operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
<i>SelfKind</i>	(Kind of arguments.)
<i>SelfSize</i>	(Number of elements.)
<i>SelfCrtp</i>	(Derived CRTP class.)
<i>SelfType</i>	(Data type.)
<i>SelfParameters</i>	(List of parameters.)

Parameters

in	<i>lhs</i>	Left-hand side.
in	<i>rhs</i>	Right-hand side.

Returns

Boolean copy.

6.46.4.13 template<typename Kind, unsigned int Size, template< typename, Kind... > class Crtp, typename Type, Kind... Parameters> template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind... > class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class > SelfCrtp<bool, SelfParameters...> operator== (const OtherType & *lhs*, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> & *rhs*) [friend]

Equal to with *lhs* value.

Applies the equal to operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
<i>SelfKind</i>	(Kind of arguments.)
<i>SelfSize</i>	(Number of elements.)
<i>SelfCrtp</i>	(Derived CRTP class.)
<i>SelfType</i>	(Data type.)
<i>SelfParameters</i>	(List of parameters.)

Parameters

in	<i>lhs</i>	Left-hand side.
in	<i>rhs</i>	Right-hand side.

Returns

Boolean copy.

6.46.4.14 template<typename Kind, unsigned int Size, template< typename, Kind... > class Crtp, typename Type, Kind... Parameters> template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind... > class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class > SelfCrtp<bool, SelfParameters...> operator> (const OtherType & *lhs*, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> & *rhs*) [friend]

Greater than with *lhs* value.

Applies the greater than operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
<i>SelfKind</i>	(Kind of arguments.)
<i>SelfSize</i>	(Number of elements.)
<i>SelfCrtp</i>	(Derived CRTP class.)
<i>SelfType</i>	(Data type.)
<i>SelfParameters</i>	(List of parameters.)

Parameters

in	<i>lhs</i>	Left-hand side.
in	<i>rhs</i>	Right-hand side.

Returns

Boolean copy.

6.46.4.15 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type, Kind... Parameters> template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class > SelfCrtp<bool, SelfParameters...> operator>= (const OtherType & *lhs*, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> & *rhs*) [friend]

Greater than or equal to with *lhs* value.

Applies the greater than or equal to operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
<i>SelfKind</i>	(Kind of arguments.)
<i>SelfSize</i>	(Number of elements.)
<i>SelfCrtp</i>	(Derived CRTP class.)
<i>SelfType</i>	(Data type.)
<i>SelfParameters</i>	(List of parameters.)

Parameters

in	<i>lhs</i>	Left-hand side.
in	<i>rhs</i>	Right-hand side.

Returns

Boolean copy.

6.46.4.16 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type, Kind... Parameters> template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class > SelfCrtp<OtherType, SelfParameters...> operator>> (const OtherType & *lhs*, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> & *rhs*) [friend]

Bitwise right shift with *lhs* value.

Applies the bitwise right shift operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
<i>SelfKind</i>	(Kind of arguments.)
<i>SelfSize</i>	(Number of elements.)
<i>SelfCrtp</i>	(Derived CRTP class.)
<i>SelfType</i>	(Data type.)
<i>SelfParameters</i>	(List of parameters.)

Parameters

in	<i>lhs</i>	Left-hand side.
in	<i>rhs</i>	Right-hand side.

Returns

Copy.

6.46.4.17 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type, Kind... Parameters> template<typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters> std::istream& operator>> (std::istream & *lhs*, StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> & *rhs*) [friend]

[Input](#) stream operator.

Fills each element from the stream.

Template Parameters

<i>SelfKind</i>	(Kind of arguments.)
<i>SelfSize</i>	(Number of elements.)
<i>SelfCrtp</i>	(Derived CRTP class.)
<i>SelfType</i>	(Data type.)
<i>SelfParameters</i>	(List of parameters.)

Parameters

in, out	<i>lhs</i>	Left-hand side stream.
in	<i>rhs</i>	Right-hand side container.

Returns

[Input](#) stream.

6.46.4.18 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type, Kind... Parameters> template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class > SelfCrtp<typename std::common_type<SelfType, OtherType>::type, SelfParameters...> operator^ (const OtherType & *lhs*, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> & *rhs*) [friend]

Bitwise XOR with *lhs* value.

Applies the bitwise XOR operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
<i>SelfKind</i>	(Kind of arguments.)
<i>SelfSize</i>	(Number of elements.)
<i>SelfCrtp</i>	(Derived CRTP class.)
<i>SelfType</i>	(Data type.)
<i>SelfParameters</i>	(List of parameters.)

Parameters

in	<i>lhs</i>	Left-hand side.
in	<i>rhs</i>	Right-hand side.

Returns

Common type copy.

6.46.4.19 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type, Kind... Parameters> template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class > SelfCrtp<typename std::common_type<SelfType, OtherType>::type, SelfParameters...> operator| (const OtherType & *lhs*, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> & *rhs*) [friend]

Bitwise OR with *lhs* value.

Applies the bitwise OR operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
<i>SelfKind</i>	(Kind of arguments.)
<i>SelfSize</i>	(Number of elements.)
<i>SelfCrtp</i>	(Derived CRTP class.)
<i>SelfType</i>	(Data type.)
<i>SelfParameters</i>	(List of parameters.)

Parameters

in	<i>lhs</i>	Left-hand side.
in	<i>rhs</i>	Right-hand side.

Returns

Common type copy.

6.46.4.20 template<typename Kind, unsigned int Size, template< typename, Kind...> class Crtp, typename Type, Kind... Parameters> template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class > SelfCrtp<bool, SelfParameters...> operator|| (const OtherType & *lhs*, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> & *rhs*) [friend]

Logical OR with *lhs* value.

Applies the logical OR operator to each element.

Template Parameters

<i>OtherType</i>	(Other data type.)
<i>SelfKind</i>	(Kind of arguments.)
<i>SelfSize</i>	(Number of elements.)
<i>SelfCrtp</i>	(Derived CRTP class.)
<i>SelfType</i>	(Data type.)
<i>SelfParameters</i>	(List of parameters.)

Parameters

in	<i>lhs</i>	Left-hand side.
in	<i>rhs</i>	Right-hand side.

Returns

Boolean copy.

The documentation for this class was generated from the following file:

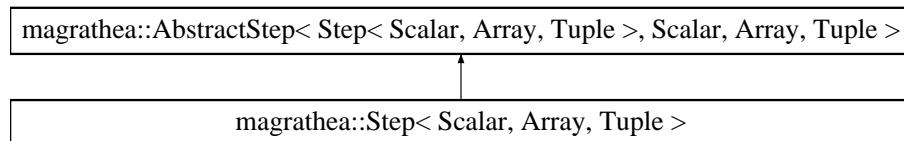
- /data/home/mbreton/magrathea_pathfinder/src/magrathea/staticvectorizer.h

6.47 magrathea::Step< Scalar, Array, Tuple > Exception Template Reference

Basic implementation of an evolution step.

```
#include <step.h>
```

Inheritance diagram for magrathea::Step< Scalar, Array, Tuple >:



Public Member Functions

Lifecycle

- template<class... Misc>
[Step](#) (Misc &&...misc)
Explicit generic constructor.

Static Public Member Functions

Test

- static int [example](#) ()
Example function.

Public Attributes

- using [operator](#) = typedef

Additional Inherited Members

6.47.1 Detailed Description

```
template<class Scalar = unsigned int, class Array = std::array<double, 0>, class Tuple = std::tuple<>>exception magrathea::Step< Scalar, Array, Tuple >
```

Basic implementation of an evolution step.

This class is the direct derivation of [AbstractStep](#). It provides the most basic and generic contents object without adding new functionalities to the abstract class. It can be used in most cases as a generic container for evolution data.

Template Parameters

<i>Scalar</i>	Scalar type of id.
<i>Array</i>	Array type of core quantities.
<i>Tuple</i>	Tuple type of extra quantities.

6.47.2 Constructor & Destructor Documentation

6.47.2.1 `template<class Scalar , class Array , class Tuple > template<class... Misc> magrathea::Step< Scalar, Array, Tuple >::Step(Misc &... misc) [inline], [explicit]`

Explicit generic constructor.

Provides a generic interface to all constructors of the base class.

Template Parameters

<i>Misc</i>	(Miscellaneous types.)
-------------	------------------------

Parameters

<i>in</i>	<i>misc</i>	Miscellaneous arguments.
-----------	-------------	--------------------------

6.47.3 Member Function Documentation

6.47.3.1 `template<class Scalar , class Array , class Tuple > int magrathea::Step< Scalar, Array, Tuple >::example() [static]`

Example function.

Tests and demonstrates the use of [Step](#).

Returns

0 if no error.

6.47.4 Member Data Documentation

6.47.4.1 `template<class Scalar = unsigned int, class Array = std::array<double, 0>, class Tuple = std::tuple<>> using magrathea::Step< Scalar, Array, Tuple >::operator =`

The documentation for this exception was generated from the following file:

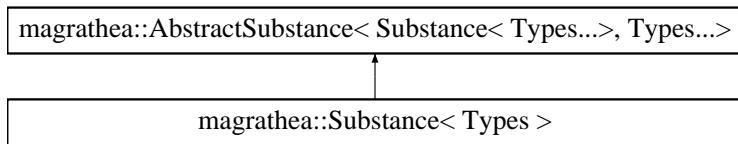
- /data/home/mbreton/magrathea_pathfinder/src/magrathea/[step.h](#)

6.48 magrathea::Substance< Types > Exception Template Reference

Basic implementation of geometrical substance.

```
#include <substance.h>
```

Inheritance diagram for magrathea::Substance< Types >:



Public Member Functions

Lifecycle

- template<class... Types> exception **Substance** (Misc &&...misc)
Explicit generic constructor.

Static Public Member Functions

Test

- static int **example** ()
Example function.

Public Attributes

- using **operator** = typedef

Additional Inherited Members

6.48.1 Detailed Description

template<class... Types> exception **magrathea::Substance< Types >**

Basic implementation of geometrical substance.

This class is the direct derivation of **AbstractSubstance**. It provides the most basic and generic substance object without adding new functionalities to the abstract class. It can be used in most cases as a generic container of groups of quantities.

Template Parameters

Types	Variadic list of components types.
--------------	------------------------------------

6.48.2 Constructor & Destructor Documentation

6.48.2.1 template<class... Types> template<class... Misc> **magrathea::Substance< Types >::Substance** (Misc &&... misc) [inline], [explicit]

Explicit generic constructor.

Provides a generic interface to all constructors of the base class.

Template Parameters

Misc	(Miscellaneous types.)
-------------	------------------------

Parameters

in	misc	Miscellaneous arguments.
----	------	--------------------------

6.48.3 Member Function Documentation**6.48.3.1 template<class... Types> int magrathea::Substance< Types >::example() [static]**

Example function.

Tests and demonstrates the use of [Substance](#).

Returns

0 if no error.

6.48.4 Member Data Documentation**6.48.4.1 template<class... Types> using magrathea::Substance< Types >::operator =**

The documentation for this exception was generated from the following file:

- /data/home/mbreton/mgrathea_pathfinder/src/mgrathea/[substance.h](#)

6.49 mgrathea::Timer< Type, Period, Clock > Exception Template Reference

A timer to manage time measurements and benchmarks.

```
#include <timer.h>
```

Public Member Functions**Lifecycle**

- template<class Duration = Type, class ReferenceTimePoint = std::true_type, class BeginningTimePoint = std::true_type, class EndingTimePoint = std::true_type>
Timer (const bool running0=false, const Duration &record0=Duration(), const ReferenceTimePoint &reference0=ReferenceTimePoint(), const BeginningTimePoint &beginning0=BeginningTimePoint(), const EndingTimePoint &ending0=EndingTimePoint())
Implicit contents constructor.
- template<typename OtherType , class OtherPeriod >
Timer (const [Timer](#)< OtherType, OtherPeriod, Clock > &source)
Implicit conversion constructor.

Operators

- template<typename OtherType , class OtherPeriod >
Timer< Type, Period, Clock > &[operator=](#) (const [Timer](#)< OtherType, OtherPeriod, Clock > &rhs)
Conversion assignment operator.
- Type [operator\(\)](#) () const
Total duration extraction operator.

Assignment

- template<class Duration = Type, class ReferenceTimePoint = std::true_type, class BeginningTimePoint = std::true_type, class EndingTimePoint = std::true_type>
`Timer< Type, Period, Clock >` & `assign` (const bool running0=false, const Duration &record0=Duration(), const ReferenceTimePoint &reference0=ReferenceTimePoint(), const BeginningTimePoint &beginning0=BeginningTimePoint(), const EndingTimePoint &ending0=EndingTimePoint())
Contents assignment.
- template<typename OtherType , class OtherPeriod >
`Timer< Type, Period, Clock >` & `assign` (const `Timer< OtherType, OtherPeriod, Clock >` &source)
Conversion assignment.

Management

- `Timer< Type, Period, Clock >` `copy () const`
Copy.
- template<typename OtherType = Type, class OtherPeriod = Period>
`Timer< OtherType, OtherPeriod,`
`Clock >` `cast () const`
Cast.

Getters

- const bool & `running () const`
Get whether the timer is running.
- const `Clock::duration & record () const`
Get the saved duration.
- const `Clock::time_point & reference () const`
Get the reference time point.
- const `Clock::time_point & beginning () const`
Get the beginning time point.
- const `Clock::time_point & ending () const`
Get the ending time point.

Actions

- `std::chrono::duration< Type,`
`Period >` `reset ()`
Reset timer.
- `std::chrono::duration< Type,`
`Period >` `start ()`
Start timer.
- `std::chrono::duration< Type,`
`Period >` `stop ()`
Stop timer.

Measurement

- `std::chrono::duration< Type,`
`Period >` `current () const`
Duration since last start.
- `std::chrono::duration< Type,`
`Period >` `total () const`
Total duration.
- `std::chrono::duration< Type,`
`Period >` `real () const`
Real duration.

Static Public Member Functions

Utilities

- template<class Duration = Type, typename Counter = unsigned long long int>
`static std::chrono::duration< Type, Period > wait (const Duration &delay=Duration(1), Counter &&counter=Counter())`
Wait a certain time.
- template<typename Counter , class Function , class... Args>
`static std::chrono::duration< Type, Period > benchmark (const Counter &counter, Function &&f, Args &&...args)`
Benchmark a function.

Test

- static int `example ()`
Example function.

Protected Attributes

Data members

- bool `_running`
Flag to indicate whether a measurement is running or not.
- `Clock::duration _record`
Internal backup of duration.
- `Clock::time_point _reference`
Reference time point for measurements.
- `Clock::time_point _beginning`
Beginning time point for measurements.
- `Clock::time_point _ending`
Ending time point for measurements.

Friends

Stream

- template<typename SelfType , class SelfPeriod , class SelfClock >
`std::ostream & operator<< (std::ostream &lhs, const Timer< SelfType, SelfPeriod, SelfClock > &rhs)`
Output stream operator.

6.49.1 Detailed Description

`template<typename Type = double, class Period = std::chrono::seconds::period, class Clock = std::chrono::steady_clock>exception magrathea::Timer< Type, Period, Clock >`

A timer to manage time measurements and benchmarks.

Provides a wrapper of `std::chrono` for an easy use and basic operations needed by time execution management. It has two internal times points : one to mark the beginning of a measurement, and one to mark the end of the current measurement. It has also a reference time point to evaluate real elasped time.

Template Parameters

<code>Type</code>	Duration representation type.
<code>Period</code>	Standard ratio representing the tick period.
<code>Clock</code>	Internal clock type.

6.49.2 Constructor & Destructor Documentation

6.49.2.1 `template<typename Type , class Period , class Clock > template<class Duration , class ReferenceTimePoint , class BeginningTimePoint , class EndingTimePoint > magrathea::Timer< Type, Period, Clock >::Timer (const bool running0 = false, const Duration & record0 = Duration(), const ReferenceTimePoint & reference0 = ReferenceTimePoint(), const BeginningTimePoint & beginning0 = BeginningTimePoint(), const EndingTimePoint & ending0 = EndingTimePoint()) [inline]`

Implicit contents constructor.

Provides a construction from every single parameter of the timer : the current record of duration, the time of reference, the time of the last start, the time of the last stop, and whether the timer is running. If no reference is provided, the current time is set. If no beginning time is provided, it is set to the reference time. If no ending time is provided, it is set to the beginning.

Template Parameters

<i>Duration</i>	(Standard duration type or arithmetic type.)
<i>ReferenceTimePoint</i>	(Time point type of the reference.)
<i>BeginningTimePoint</i>	(Time point type of the beginning time.)
<i>EndingTimePoint</i>	(Time point type of the ending time.)

Parameters

in	<i>running0</i>	Input value of whether the timer is running.
in	<i>record0</i>	Input value of the backup of duration.
in	<i>reference0</i>	Input value of the reference time.
in	<i>beginning0</i>	Input value of the beginning time.
in	<i>ending0</i>	Input value of the ending time.

6.49.2.2 `template<typename Type , class Period , class Clock > template<typename OtherType , class OtherPeriod > magrathea::Timer< Type, Period, Clock >::Timer (const Timer< OtherType, OtherPeriod, Clock > & source) [inline]`

Implicit conversion constructor.

Provides a construction from a timer of another type.

Template Parameters

<i>OtherType</i>	(Other duration representation type.)
<i>OtherPeriod</i>	(Other standard ratio representing the tick period.)

Parameters

in	<i>source</i>	Source of the copy.
----	---------------	---------------------

6.49.3 Member Function Documentation

6.49.3.1 `template<typename Type , class Period , class Clock > template<class Duration , class ReferenceTimePoint , class BeginningTimePoint , class EndingTimePoint > Timer< Type, Period, Clock > & magrathea::Timer< Type, Period, Clock >::assign (const bool running0 = false, const Duration & record0 = Duration(), const ReferenceTimePoint & reference0 = ReferenceTimePoint(), const BeginningTimePoint & beginning0 = BeginningTimePoint(), const EndingTimePoint & ending0 = EndingTimePoint()) [inline]`

[Contents](#) assignment.

Assigns contents from every single parameter of the timer : the current record of duration, the time of reference, the time of the last start, the time of the last stop, and whether the timer is running. If no reference is provided, the current time is set. If no beginning time is provided, it is set to the reference time. If no ending time is provided, it is set to the beginning.

Template Parameters

<i>Duration</i>	(Standard duration type or arithmetic type.)
<i>ReferenceTimePoint</i>	(Time point type of the reference.)
<i>BeginningTimePoint</i>	(Time point type of the beginning time.)
<i>EndingTimePoint</i>	(Time point type of the ending time.)

Parameters

in	<i>running0</i>	Input value of whether the timer is running.
in	<i>record0</i>	Input value of the backup of duration.
in	<i>reference0</i>	Input value of the reference time.
in	<i>beginning0</i>	Input value of the beginning time.
in	<i>ending0</i>	Input value of the ending time.

Returns

Self reference.

6.49.3.2 template<typename Type , class Period , class Clock > template<typename OtherType , class OtherPeriod > Timer< Type, Period, Clock > & magrathea::Timer< Type, Period, Clock >::assign (const Timer< OtherType, OtherPeriod, Clock > & source) [inline]

Conversion assignment.

Assigns the contents from a timer of another type.

Template Parameters

<i>OtherType</i>	(Other duration representation type.)
<i>OtherPeriod</i>	(Other standard ratio representing the tick period.)

Parameters

in	<i>source</i>	Source of the copy.
----	---------------	---------------------

Returns

Self reference.

6.49.3.3 template<typename Type , class Period , class Clock > const Clock::time_point & magrathea::Timer< Type, Period, Clock >::beginning () const [inline]

Get the beginning time point.

Returns the beginning time point, which is generally the time point of the last start.

Returns

Beginning time point.

6.49.3.4 `template<typename Type , class Period , class Clock > template<typename Counter , class Function , class... Args> std::chrono::duration< Type, Period > magrathea::Timer< Type, Period, Clock >::benchmark (const Counter & counter, Function && f, Args &&... args) [inline], [static]`

Benchmark a function.

Executes the provided function in a loop and computes the total time needed to run it. The call uses a volatile temporary to prevent null statement optimization, but some compilers may manage to optimize that. The results are not guaranteed to be exact and should be checked with a real benchmarking suite. The returned time correspond to the time of function execution plus the time of the copy.

Template Parameters

<i>Counter</i>	(Type that can be incremented.)
<i>Function</i>	(Function type : Something (Args...).)
<i>Args</i>	(Arguments types.)

Parameters

in	<i>counter</i>	Number of loops to do.
in	<i>f</i>	Function object Something (Args...).
in	<i>args</i>	Arguments of the function.

Returns

Total duration of loop execution.

6.49.3.5 `template<typename Type , class Period , class Clock > template<typename OtherType , class OtherPeriod > Timer< OtherType, OtherPeriod, Clock > magrathea::Timer< Type, Period, Clock >::cast () const [inline]`

Cast.

Casts the timer to another timer type with another period.

Template Parameters

<i>OtherType</i>	Other duration representation type.
<i>OtherPeriod</i>	Other standard ratio representing the tick period.

Returns

Copy.

6.49.3.6 `template<typename Type , class Period , class Clock > Timer< Type, Period, Clock > magrathea::Timer< Type, Period, Clock >::copy () const [inline]`

Copy.

Returns a copy of the timer.

Returns

Copy.

6.49.3.7 template<typename Type , class Period , class Clock > std::chrono::duration< Type, Period >
magrathea::Timer< Type, Period, Clock >::current() const [inline]

Duration since last start.

Computes the duration since last start : if the timer is still running, it computes the difference between the call time and the last start time and if the timer is not running it returns the difference between the last start and the last stop.

Returns

Current duration.

6.49.3.8 template<typename Type , class Period , class Clock > const Clock::time_point & magrathea::Timer< Type, Period, Clock >::ending() const [inline]

Get the ending time point.

Returns the ending time point, which is generally the time point of the last stop. If the timer is running, then the ending time point is equal to the last start time point.

Returns

Ending time point.

6.49.3.9 template<typename Type , class Period , class Clock > int magrathea::Timer< Type, Period, Clock >::example() [static]

Example function.

Tests and demonstrates the use of [Timer](#).

Returns

0 if no error.

6.49.3.10 template<typename Type , class Period , class Clock > Type magrathea::Timer< Type, Period, Clock >::operator()() const [inline]

Total duration extraction operator.

Computes the total elapsed duration between all starts and stops since the last reset and convert it to an arithmetic type.

Returns

Total duration.

6.49.3.11 template<typename Type , class Period , class Clock > template<typename OtherType , class OtherPeriod >
Timer< Type, Period, Clock > & magrathea::Timer< Type, Period, Clock >::operator=(const Timer< OtherType, OtherPeriod, Clock > & rhs) [inline]

Conversion assignment operator.

Assigns the contents from a timer of another type.

Template Parameters

<i>OtherType</i>	(Other duration representation type.)
<i>OtherPeriod</i>	(Other standard ratio representing the tick period.)

Parameters

in	<i>rhs</i>	Right-hand side.
----	------------	------------------

Returns

Self reference.

6.49.3.12 template<typename Type , class Period , class Clock > std::chrono::duration< Type, Period > magrathea::Timer< Type, Period, Clock >::real() const [inline]

Real duration.

Computes the real duration since the last reset without considering any start and stop.

Returns

Real duration.

6.49.3.13 template<typename Type , class Period , class Clock > const Clock::duration & magrathea::Timer< Type, Period, Clock >::record() const [inline]

Get the saved duration.

Returns the value of the saved duration which is the total duration saved during the last stop.

Returns

Duration.

6.49.3.14 template<typename Type , class Period , class Clock > const Clock::time_point & magrathea::Timer< Type, Period, Clock >::reference() const [inline]

Get the reference time point.

Returns the reference time point, which is generally the time point of the last reset.

Returns

Reference time point.

6.49.3.15 template<typename Type , class Period , class Clock > std::chrono::duration< Type, Period > magrathea::Timer< Type, Period, Clock >::reset() [inline]

Reset timer.

Resets the timer : all time points are set to the current time, the duration is set to zero, and the timer is set off.

Returns

Current duration, which is equal to zero.

6.49.3.16 `template<typename Type , class Period , class Clock > const bool & magrathea::Timer< Type, Period, Clock >::running() const [inline]`

Get whether the timer is running.

Returns true if the timer is running, false if it was stopped.

Returns

Running status.

6.49.3.17 `template<typename Type , class Period , class Clock > std::chrono::duration< Type, Period > magrathea::Timer< Type, Period, Clock >::start() [inline]`

Start timer.

Starts the timer for a new measurement. If the timer is already running, the previous state is erased.

Returns

Current duration, which is equal to zero.

6.49.3.18 `template<typename Type , class Period , class Clock > std::chrono::duration< Type, Period > magrathea::Timer< Type, Period, Clock >::stop() [inline]`

Stop timer.

Stops the timer, adds the duration to the total one, and returns the time since the previous start. If the timer is already not running, nothing is done.

Returns

Current duration.

6.49.3.19 `template<typename Type , class Period , class Clock > std::chrono::duration< Type, Period > magrathea::Timer< Type, Period, Clock >::total() const [inline]`

Total duration.

Computes the total elapsed duration between all starts and stops since the last reset.

Returns

Total duration.

6.49.3.20 `template<typename Type , class Period , class Clock > template<class Duration , typename Counter > std::chrono::duration< Type, Period > magrathea::Timer< Type, Period, Clock >::wait(const Duration & delay = Duration(1), Counter && counter = Counter()) [inline], [static]`

Wait a certain time.

Loops over time in order to delay some operation. If a counter is passed, it is incremented at each loop. The loop ends when the elapsed time is greater or equal to the specified delay. This delay can be a number or a standard duration. Internally, it is casted to the timer type, and then to a high precision clock. At the end of the function the total duration is casted back from the high precision clock to the timer duration type.

Template Parameters

<i>Duration</i>	(Standard duration type or arithmetic type.)
<i>Counter</i>	(Type that can be incremented.)

Parameters

<i>in</i>	<i>delay</i>	Duration to wait.
<i>in,out</i>	<i>counter</i>	Incremented counter at each loop step.

Returns

The real elapsed time in the function.

6.49.4 Friends And Related Function Documentation

6.49.4.1 `template<typename Type = double, class Period = std::chrono::seconds::period, class Clock = std::chrono::steady_clock> template<typename SelfType , class SelfPeriod , class SelfClock > std::ostream& operator<< (std::ostream & lhs, const Timer< SelfType, SelfPeriod, SelfClock > & rhs) [friend]`

[Output](#) stream operator.

Prints out the total duration.

Template Parameters

<i>SelfType</i>	(Duration representation type.)
<i>SelfPeriod</i>	(Standard ratio representing the tick period.)
<i>SelfClock</i>	(Internal clock type.)

Parameters

<i>in,out</i>	<i>lhs</i>	Left-hand side stream.
<i>in</i>	<i>rhs</i>	Right-hand side timer.

Returns

[Output](#) stream.

6.49.5 Member Data Documentation

6.49.5.1 `template<typename Type = double, class Period = std::chrono::seconds::period, class Clock = std::chrono::steady_clock> Clock::time_point magrathea::Timer< Type, Period, Clock >::_beginning [protected]`

Beginning time point for measurements.

6.49.5.2 `template<typename Type = double, class Period = std::chrono::seconds::period, class Clock = std::chrono::steady_clock> Clock::time_point magrathea::Timer< Type, Period, Clock >::_ending [protected]`

Ending time point for measurements.

6.49.5.3 template<typename Type = double, class Period = std::chrono::seconds::period, class Clock = std::chrono::steady_clock> Clock::duration magrathea::Timer< Type, Period, Clock >::_record [protected]

Internal backup of duration.

6.49.5.4 template<typename Type = double, class Period = std::chrono::seconds::period, class Clock = std::chrono::steady_clock> Clock::time_point magrathea::Timer< Type, Period, Clock >::_reference [protected]

Reference time point for measurements.

6.49.5.5 template<typename Type = double, class Period = std::chrono::seconds::period, class Clock = std::chrono::steady_clock> bool magrathea::Timer< Type, Period, Clock >::_running [protected]

Flag to indicate whether a measurement is running or not.

The documentation for this exception was generated from the following file:

- /data/home/mbreton/magrathea_pathfinder/src/magrathea/timer.h

6.50 TReadHDF5 Class Reference

```
#include <TReadHDF5.h>
```

Public Member Functions

- void [displayAllInfos](#) (const std::string &fileName) const
Prints informations.
- template<typename Type1 >
void [fillVectors_part](#) (const double &fraction, const std::string &fileName, const std::string &fileSide, const std::string &output_name1, std::vector< Type1 > &output1)
Get data from part cones.

Static Public Member Functions

- static hid_t [h5t_native](#) (const int N)
Converts Type to the corresponding hid_t.
- static hid_t [h5t_native](#) (const unsigned int N)
Converts Type to the corresponding hid_t.
- static hid_t [h5t_native](#) (const float N)
Converts Type to the corresponding hid_t.
- static hid_t [h5t_native](#) (const double N)
Converts Type to the corresponding hid_t.
- static hid_t [h5t_native](#) (const unsigned long int N)
Converts Type to the corresponding hid_t.
- template<typename T >
static hid_t [h5t_native](#) (void)
Converts Type to the corresponding hid_t.
- static void [pos_from_index_fullsky](#) (const std::string &str, const std::vector< int > &nctab, const std::vector< float > &dimtab, const float &cubesize, std::array< double, 3 > &point111)

- static void **pos_from_index_narrow** (const std::string &str, const std::vector< int > &nctab, const std::vector< float > &dimtab, const float &cubesize, const double &thetay, const double &thetaz, std::array< double, 3 > &point111)
 - Get position from index.*
- template<typename Type >
 - static void **get_data_from_dataset** (const hid_t &gid, const std::string &output_name, std::vector< Type > &output)
- *Get data from dataset.*
- template<typename Type , typename... String, typename... Vector>
 - static void **get_data_from_dataset** (const hid_t &gid, const std::string &output_name, std::vector< Type > &output, const String &...output_names, Vector &...outputs)
- *Get data from dataset.*
- template<typename Integer >
 - static void **cellsPerLevels** (const std::string &fileName, std::vector< Integer > &count)

Number of cells.
- template<class Parameter , typename Integer , class Conic >
 - static void **cellsAndCubesPerLevels** (const Parameter ¶meters, const std::string &fileName, std::vector< Integer > &count, std::vector< std::vector< std::string > > &cubeNumber, const double &thetay, const double &thetaz, const Conic &conic)
- *Get cells per level.*
- template<typename Type >
 - static void **getAttribute** (const std::string &fileName, const std::string &attributeName, Type &attributeValue)
- *Get attribute.*
- template<typename Type >
 - static void **getAttribute** (const std::string &fileName, const std::string &attributeName, std::vector< Type > &attributeValue)
- *Get attribute.*
- template<typename Type >
 - static void **getAttribute** (const std::string &fileName, const std::string &attributeName1, const std::string &attributeName2, Type &attributeValue)
- *Get attribute.*
- template<typename Type >
 - static void **getAttribute** (const std::string &fileName, const std::string &attributeName1, const std::string &attributeName2, std::vector< Type > &attributeValue)
- *Get attribute.*
- template<typename Type >
 - static void **fillVectors_grav** (const std::string &fileName, const unsigned int &levelMin, const unsigned int &levelMax, const std::string &output_name, std::vector< Type > &output)
- *Get data from grav cones.*
- template<typename Type , typename... String, typename... Vector>
 - static void **fillVectors_grav** (const std::string &fileName, const unsigned int &levelMin, const unsigned int &levelMax, const std::vector< std::string > &cubeNumber, const std::string &output_name, std::vector< Type > &output, const String &...output_names, Vector &...outputs)
- *Get data from grav cones.*
- template<typename Type >
 - static void **fillVectors_grav** (const std::string &fileName, const unsigned int &levelMin, const unsigned int &levelMax, const std::vector< std::string > &cubeNumber, const std::string &output_name, std::vector< Type > &output, const String &...output_names, Vector &...outputs)
- *Get data from grave cones.*
- template<typename Type , typename... String, typename... Vector>
 - static void **fillVectors_grav** (const std::string &fileName, const unsigned int &levelMin, const unsigned int &levelMax, const std::vector< std::string > &cubeNumber, const std::string &output_name, std::vector< Type > &output, const String &...output_names, Vector &...outputs)

- Get data from grav cones.*
- template<typename Type >
`static void fillVectors_part (const std::string &fileName, const std::string &fileSide, const std::string &output_name, std::vector< Type > &output)`
- Get data from part cones.*
- template<typename Type , typename... String, typename... Vector>
`static void fillVectors_part (const std::string &fileName, const std::string &fileSide, const std::string &output_name, std::vector< Type > &output, const String &...output_names, Vector &...outputs)`
- Get data from part cones.*
- template<typename Type >
`static void fillVectors_part (const double &fraction, const std::string &fileName, const std::string &fileSide, const std::string &output_name, std::vector< Type > &output)`
 - template<typename Type1 , typename Type2 >
`static void fillVectors_part (const double &fraction, const std::string &fileName, const std::string &fileSide, const std::string &output_name1, std::vector< Type1 > &output1, const std::string &output_name2, std::vector< Type2 > &output2)`
- Get data from part cones.*
- template<typename Type1 , typename Type2 , typename Type3 >
`static void fillVectors_part (const double &fraction, const std::string &fileName, const std::string &fileSide, const std::string &output_name1, std::vector< Type1 > &output1, const std::string &output_name2, std::vector< Type2 > &output2, const std::string &output_name3, std::vector< Type3 > &output3)`
- Get data from part cones.*
- template<class Parameter , typename Type , class Conic >
`static void fillVectors_part (const Parameter ¶meters, const std::string &fileName, const double &thetay, const double &thetaz, const Conic &conic, const std::string &fileSide, const std::string &output_name, std::vector< Type > &output)`
- Get data from part cones.*
- template<class Parameter , typename Type , typename... String, typename... Vector, class Conic >
`static void fillVectors_part (const Parameter ¶meters, const std::string &fileName, const double &thetay, const double &thetaz, const Conic &conic, const std::string &fileSide, const std::string &output_name, std::vector< Type > &output, const String &...output_names, Vector &...outputs)`
- Get data from part cones.*

6.50.1 Member Function Documentation

6.50.1.1 template<class Parameter , typename Integer , class Conic > void TReadHDF5::cellsAndCubesPerLevels (const Parameter & parameters, const std::string & fileName, std::vector< Integer > & count, std::vector< std::vector< std::string > > & cubeNumber, const double & thetay, const double & thetaz, const Conic & conic) [static]

Get cells per level.

Get cells per level.

Template Parameters

<i>Parameter</i>	Parameter type.
<i>Integer</i>	Count type
<i>Conic</i>	Cone type

Parameters

<i>in</i>	<i>parameters</i>	Parameter structure.
<i>in</i>	<i>fileName</i>	File name.
<i>in,out</i>	<i>count</i>	Number of cells per level
<i>in</i>	<i>cubeNumber</i>	Name number of cubes per level.
<i>in</i>	<i>thetay</i>	Semi-angle for solid angle in direction y

in	<i>thetaZ</i>	Semi-angle for solid angle in direction z
in	<i>conic</i>	Cone.

6.50.1.2 template<typename Integer > void TReadHDF5::cellsPerLevels (const std::string & *fileName*, std::vector< Integer > & *count*) [static]

Number of cells.

Number of cells per level

Template Parameters

<i>Integer</i>	Count type.
----------------	-------------

Parameters

in	<i>fileName</i>	File name.
in, out	<i>count</i>	Number of cells per level.

6.50.1.3 void TReadHDF5::displayAllInfos (const std::string & *fileName*) const

Prints informations.

Prints the following informations on the HDF5 file :

- File name
- Group names
- Number of subgroups

Parameters

in	<i>fileName</i>	File list.
----	-----------------	------------

6.50.1.4 template<typename Type > void TReadHDF5::fillVectors_grav (const std::string & *fileName*, const unsigned int & *levelMin*, const unsigned int & *levelMax*, const std::string & *output_name*, std::vector< Type > & *output*) [static]

Get data from grav cones.

Get list of values of a dataset.

Template Parameters

<i>Type</i>	Data type.
-------------	------------

Parameters

in	<i>fileName</i>	File name.
in	<i>levelMin</i>	int Minimum level.
in	<i>levelMax</i>	int Maximum level.
in	<i>output_name</i>	Data name.
in, out	<i>output</i>	Data vector.

6.50.1.5 template<typename Type , typename... String, typename... Vector> void TReadHDF5::fillVectors_grav (const std::string & *fileName*, const unsigned int & *levelMin*, const unsigned int & *levelMax*, const std::string & *output_name*, std::vector< Type > & *output*, const String &... *output_names*, Vector &... *outputs*) [static]

Get data from grav cones.

Get list of values of a dataset.

Template Parameters

Type	Data type.
------	------------

Parameters

in	<i>fileName</i>	File name.
in	<i>levelMin</i>	int Minimum level.
in	<i>levelMax</i>	int Maximum level.
in	<i>output_name</i>	Data name.
in,out	<i>output</i>	Data vector.
in	<i>output_names</i>	Variadic Data names.
in,out	<i>outputs</i>	Variadic Data vectors.

6.50.1.6 template<typename Type > void TReadHDF5::fillVectors_grav (const std::string & *fileName*, const unsigned int & *levelMin*, const unsigned int & *levelMax*, const std::vector< std::string > & *cubeNumber*, const std::string & *output_name*, std::vector< Type > & *output*) [static]

Get data from grave cones.

Get list of values of a dataset of a cube that intersects the cone GIVEN the list of cubes as inputs

Template Parameters

Type	Data type.
------	------------

Parameters

in	<i>fileName</i>	File name.
in	<i>levelMin</i>	Minimum level.
in	<i>levelMax</i>	Maximum level.
in,out	<i>cubeNumber</i>	Name of cubes that intersects the cone.
in	<i>output_name</i>	Data name.
in,out	<i>output</i>	Data vector.

6.50.1.7 template<typename Type , typename... String, typename... Vector> void TReadHDF5::fillVectors_grav (const std::string & *fileName*, const unsigned int & *levelMin*, const unsigned int & *levelMax*, const std::vector< std::string > & *cubeNumber*, const std::string & *output_name*, std::vector< Type > & *output*, const String &... *output_names*, Vector &... *outputs*) [static]

Get data from grav cones.

Get lists of values of datasets of a cube that intersects the cone GIVEN the list of cubes as inputs

Template Parameters

Type1	Data type.
Type2	Data type.

Parameters

in	<i>fileName</i>	File name.
in	<i>levelMin</i>	Minimum level.
in	<i>levelMax</i>	Maximum level.
in,out	<i>cubeNumber</i>	Name of cubes that intersects the cone.
in	<i>output_name</i>	Data name.
in,out	<i>output</i>	Data vector.
in	<i>output_names</i>	Variadic Data names.
in,out	<i>outputs</i>	Variadic Data vectors.

6.50.1.8 template<typename Type > void TReadHDF5::fillVectors_part (const std::string & *fileName*, const std::string & *fileSide*, const std::string & *output_name*, std::vector< Type > & *output*) [static]

Get data from part cones.

Get lists of values of datasets

Template Parameters

Type	dataset type.
------	---------------

Parameters

in	<i>fileName</i>	File name.
in	<i>fileSide</i>	side name('data' or 'metadata' in Raygal simulation HDF5 files).
in	<i>output_name</i>	Data name.
in,out	<i>output</i>	Data vector.

6.50.1.9 template<typename Type , typename... String, typename... Vector> void TReadHDF5::fillVectors_part (const std::string & *fileName*, const std::string & *fileSide*, const std::string & *output_name*, std::vector< Type > & *output*, const String &... *output_names*, Vector &... *outputs*) [static]

Get data from part cones.

Get lists of values of datasets

Template Parameters

Type1	Dataset type.
Type2	Dataset type.

Parameters

in	<i>fileName</i>	File name.
in	<i>fileSide</i>	side name('data' or 'metadata' in Raygal simulation HDF5 files).
in	<i>output_name</i>	Data name.
in,out	<i>output</i>	Data vector.
in	<i>output_names</i>	Variadic Data names.
in,out	<i>outputs</i>	Variadic Data vectors.

6.50.1.10 template<typename Type > static void TReadHDF5::fillVectors_part (const double & *fraction*, const std::string & *fileName*, const std::string & *fileSide*, const std::string & *output_name*, std::vector< Type > & *output*) [static]

```
6.50.1.11 template<typename Type1 , typename Type2> void TReadHDF5::fillVectors_part ( const double & fraction, const std::string & fileName, const std::string & fileSide, const std::string & output_name1, std::vector< Type1 > & output1, const std::string & output_name2, std::vector< Type2 > & output2 ) [static]
```

Get data from part cones.

Get lists of values of datasets

Template Parameters

Type1	Dataset type.
Type2	Dataset type.

Parameters

in	fraction	probability under which we take the data
in	fileName	File name.
in	fileSide	side name(data or metadata).
in	output_name1	Data name.
in,out	output1	Data vector.
in	output_name2	Variadic Data name.
in,out	output2	Variadic Data vector.

```
6.50.1.12 template<typename Type1 , typename Type2, typename Type3> void TReadHDF5::fillVectors_part ( const double & fraction, const std::string & fileName, const std::string & fileSide, const std::string & output_name1, std::vector< Type1 > & output1, const std::string & output_name2, std::vector< Type2 > & output2, const std::string & output_name3, std::vector< Type3 > & output3 ) [static]
```

Get data from part cones.

Get lists of values of datasets

Template Parameters

Type1	Dataset type.
Type2	Dataset type.
Type3	Dataset type.

Parameters

in	fraction	probability under which we take the data
in	fileName	File name.
in	fileSide	side name(data or metadata).
in	output_name1	Data name.
in,out	output1	Data vector.
in	output_name2	Data name.
in,out	output2	Data vector.
in	output_name3	Data name.
in,out	output3	Data vector.

```
6.50.1.13 template<class Parameter , typename Type, class Conic> void TReadHDF5::fillVectors_part ( const Parameter & parameters, const std::string & fileName, const double & theta_y, const double & theta_z, const Conic & conic, const std::string & fileSide, const std::string & output_name, std::vector< Type > & output ) [static]
```

Get data from part cones.

Get values of datasets

Template Parameters

<i>Parameter</i>	Parameter type.
<i>Type1</i>	Dataset type.
<i>Type2</i>	Dataset type.
<i>Conic</i>	Cone type.

Parameters

in	<i>parameters</i>	Parameter structure.
in	<i>fileName</i>	File name.
in	<i>thetay</i>	Semi-angle for solid angle in direction y
in	<i>thetaz</i>	Semi-angle for solid angle in direction z
in	<i>conic</i>	Cone .
in	<i>fileSide</i>	side name(data or metadata).
in	<i>output_name</i>	Data name.
in, out	<i>output</i>	Data vector.

```
6.50.1.14 template<class Parameter , typename Type, typename... String, typename... Vector, class Conic > void
TReadHDF5::fillVectors_part ( const Parameter & parameters, const std::string & fileName, const double & thetay,
const double & thetaz, const Conic & conic, const std::string & fileSide, const std::string & output_name,
std::vector< Type > & output, const String &... output_names, Vector &... outputs ) [static]
```

Get data from part cones.

Get values of datasets

Template Parameters

<i>Parameter</i>	Parameter type.
<i>Type1</i>	Dataset type.
<i>Type2</i>	Dataset type.
<i>Conic</i>	Cone type.

Parameters

in	<i>parameters</i>	Parameter structure.
in	<i>fileName</i>	File name.
in	<i>thetay</i>	Semi-angle for solid angle in direction y
in	<i>thetaz</i>	Semi-angle for solid angle in direction z
in	<i>conic</i>	Cone .
in	<i>fileSide</i>	side name(data or metadata).
in	<i>output_name</i>	Data name.
in, out	<i>output</i>	Data vector.
in	<i>output_names</i>	Variadic Data names.
in, out	<i>outputs</i>	Variadic Data vectors.

```
6.50.1.15 template<typename Type1 > void TReadHDF5::fillVectors_part ( const double & fraction, const std::string &
fileName, const std::string & fileSide, const std::string & output_name1, std::vector< Type1 > & output1 )
```

Get data from part cones.

Get lists of values of datasets

Template Parameters

Type1	Dataset type.
-------	---------------

Parameters

in	<i>fraction</i>	Probability under which we take the data
in	<i>fileName</i>	File name.
in	<i>fileSide</i>	side name(data or metadata).
in	<i>output_name1</i>	Data name.
in,out	<i>output1</i>	Data vector.

6.50.1.16 template<typename Type > void TReadHDF5::get_data_from_dataset (const hid_t & *gid*, const std::string & *output_name*, std::vector< Type > & *output*) [static]

Get data from dataset.

Get data from dataset

Template Parameters

Type	Type type.
------	------------

Parameters

in	<i>gid</i>	Group HDF5 id.
in	<i>output_name</i>	Data name.
in,out	<i>output</i>	Data vector.

6.50.1.17 template<typename Type , typename... String, typename... Vector> void TReadHDF5::get_data_from_dataset (const hid_t & *gid*, const std::string & *output_name*, std::vector< Type > & *output*, const String &... *output_names*, Vector &... *outputs*) [static]

Get data from dataset.

Get data from dataset

Template Parameters

Type	Type type.
------	------------

Parameters

in	<i>gid</i>	Group HDF5 id.
in	<i>output_name</i>	Data name.
in,out	<i>output</i>	Data vector.
in	<i>output_names</i>	Variadic Data names.
in,out	<i>outputs</i>	Variadic Data vectors.

6.50.1.18 template<typename Type > void TReadHDF5::getAttribute (const std::string & *fileName*, const std::string & *attributeName*, Type & *attributeValue*) [static]

Get attribute.

Get attribute.

Template Parameters

Type	Attribute type.
------	-----------------

Parameters

in	<i>fileName</i>	File name.
in	<i>attributeName</i>	Attribute name.
in, out	<i>attributeValue</i>	Value of attribute.

6.50.1.19 template<typename Type > void TReadHDF5::getAttribute (const std::string & *fileName*, const std::string & *attributeName*, std::vector< Type > & *attributeValue*) [static]

Get attribute.

Get attribute.

Template Parameters

Type	Attribute type.
------	-----------------

Parameters

in	<i>fileName</i>	File name.
in	<i>attributeName</i>	Attribute name.
in, out	<i>attributeValue</i>	Values of attribute.

6.50.1.20 template<typename Type > void TReadHDF5::getAttribute (const std::string & *fileName*, const std::string & *groupName*, const std::string & *attributeName*, Type & *attributeValue*) [static]

Get attribute.

Get attribute.

Template Parameters

Type	Attribute type.
------	-----------------

Parameters

in	<i>fileName</i>	File name.
in	<i>groupName</i>	Group name.
in	<i>attributeName</i>	Attribute name
in, out	<i>attributeValue</i>	Value of attribute.

6.50.1.21 template<typename Type > void TReadHDF5::getAttribute (const std::string & *fileName*, const std::string & *groupName*, const std::string & *attributeName*, std::vector< Type > & *attributeValue*) [static]

Get attribute.

Get attribute.

Template Parameters

Type	Attribute type.
------	-----------------

Parameters

<i>in</i>	<i>fileName</i>	File name.
<i>in</i>	<i>groupName</i>	Group name.
<i>in</i>	<i>attributeName</i>	Attribute name.
<i>in, out</i>	<i>attributeValue-Values</i>	of attribute.

6.50.1.22 hid_t TReadHDF5::h5t_native (const int N) [static]

Converts Type to the corresponding hid_t.

The HDF5 C routines use a hid_t type in order to get the data of a specified type.

Parameters

<i>in</i>	<i>N</i>	(only the type matters, not the value)
-----------	----------	--

Returns

hid_t H5T_NATIVE of the intput type

6.50.1.23 hid_t TReadHDF5::h5t_native (const unsigned int N) [static]

Converts Type to the corresponding hid_t.

The HDF5 C routines use a hid_t type in order to get the data of a specified type.

Parameters

<i>in</i>	<i>N</i>	(only the type matters, not the value)
-----------	----------	--

Returns

hid_t H5T_NATIVE of the intput type

6.50.1.24 hid_t TReadHDF5::h5t_native (const float N) [static]

Converts Type to the corresponding hid_t.

The HDF5 C routines use a hid_t type in order to get the data of a specified type.

Parameters

<i>in</i>	<i>N</i>	(only the type matters, not the value)
-----------	----------	--

Returns

hid_t H5T_NATIVE of the intput type

6.50.1.25 hid_t TReadHDF5::h5t_native (const double N) [static]

Converts Type to the corresponding hid_t.

The HDF5 C routines use a hid_t type in order to get the data of a specified type.

Parameters

in	<i>N</i>	(only the type matters, not the value)
----	----------	--

Returns

hid_t H5T_NATIVE of the intput type

6.50.1.26 hid_t TReadHDF5::h5t_native (const unsigned long int *N*) [static]

Converts Type to the corresponding hid_t.

The HDF5 C routines use a hid_t type in order to get the data of a specified type.

Parameters

in	<i>N</i>	(only the type matters, not the value)
----	----------	--

Returns

hid_t H5T_NATIVE of the intput type

6.50.1.27 template<class *T* > hid_t TReadHDF5::h5t_native (void) [static]

Converts Type to the corresponding hid_t.

The HDF5 C routines use a hid_t type in order to get the data of a specified type.

Template Parameters

<i>T</i>	Type
----------	------

Returns

hid_t H5T_NATIVE of the intput type

6.50.1.28 void TReadHDF5::pos_from_index_fullsky (const std::string & *str*, const std::vector< int > & *nctab*, const std::vector< float > & *dimtab*, const float & *cubesize*, std::array< double, 3 > & *point111*) [static]

Get position from index.

Get position from indexes in Raygal simulation fullsky data cubes

Parameters

in	<i>str</i>	Cube name in HDF5 file.
in	<i>nctab</i>	Number of cubes in data
in	<i>dimtab</i>	Dimensions of data.
in	<i>cubesize</i>	Cube size.
in, out	<i>point111</i>	Cube center position.

```
6.50.1.29 void TReadHDF5::pos_from_index_narrow ( const std::string & str, const std::vector< int > & nctab, const
std::vector< float > & dimtab, const float & cubesize, const double & thetay, const double & thetaz, std::array<
double, 3 > & point111 ) [static]
```

Get position from index.

Get position from indexes in Raygal simulation narrow data cubes

Parameters

in	<i>str</i>	Cube name in HDF5 file.
in	<i>nctab</i>	Number of cubes in every dimension.
in	<i>dimtab</i>	Dimensions in Ramses units of the parallelepiped around the cone
in	<i>cubesize</i>	Cube size.
in	<i>thetay</i>	Semi-angle for solid angle in direction y
in	<i>thetaz</i>	Semi-angle for solid angle in direction z
in,out	<i>point111</i>	Cube center position.

The documentation for this class was generated from the following file:

- /data/home/mbreton/magrathea_pathfinder/src/TReadHDF5.h

6.51 Utility Exception Reference

List of utilities for raytracing.

```
#include <utility.h>
```

Static Public Member Functions

- template<class Type >
static std::array< std::array< Type, 2 >, 2 > **invMatrix2d** (const std::array< std::array< Type, 2 >, 2 > &A)
Inverse matrix.
- template<class Type >
static std::array< std::array< Type, 3 >, 3 > **invMatrix3d** (const std::array< std::array< Type, 3 >, 3 > &A)
Inverse matrix.

Parallelization

- template<int Default = 0, typename Type , class Function , class = typename std::enable_if<(std::is_convertible< decltype(std::declval<Type>())+std::declval<Type>(), int>::value) && (!std::is_function<typename std::result_of<Function(Type)>::type>::value)>::type>
static double **parallelize** (const Type nsteps, Function &&function, const int nthreads=(Default!=0)?(Default):(std::thread::hardware_concurrency()))
Parallelize a loop.
- template<int Default = 0, typename Type , class Function , class = typename std::enable_if<(!std::is_void<decltype(std::declval<Type>())/std::declval<Type>())>::value) && (std::is_function<typename std::result_of<Function(Type)>::type>::value)>::type>
static double **parallelize** (const Type &first, const Type &last, const Type &increment, Function &&function, const int nthreads=(Default!=0)?(Default):(std::thread::hardware_concurrency()))
Parallelize an iteration over a range of values.
- template<int Default = 0, typename Iterator , class Function , class = typename std::enable_if<(!std::is_void<decltype(*std::declval<Iterator>())>::value) && (!std::is_function<typename std::result_of<Function(decltype(*std::declval<Iterator>()))>::type>::value)>::type>
static double **parallelize** (const Iterator &first, const Iterator &last, Function &&function, const int nthreads=(Default!=0)?(Default):(std::thread::hardware_concurrency()))

Parallelize an iteration over a range of iterators.

Geometry

- template<unsigned int Dimension, unsigned int Index = 0, class Vector , typename Scalar = typename std::remove_cv<typename std::remove_reference<decltype(std::declval<Vector>()[0])>::type>::type>
static constexpr Scalar **distance** (const Vector &first, const Vector &second)
Euclidian distance.
- template<unsigned int Dimension, unsigned int Index = 0, class Vector , typename Scalar = typename std::remove_cv<typename std::remove_reference<decltype(std::declval<Vector>()[0])>::type>::type>
static constexpr Scalar **dot** (const Vector &first, const Vector &second)
Dot product.
- template<unsigned int Dimension, unsigned int Index = 0, class Vector , typename Scalar = typename std::remove_cv<typename std::remove_reference<decltype(std::declval<Vector>()[0])>::type>::type>
static Vector **cross** (const Vector &first, const Vector &second)
Three dimensional cross product.
- template<unsigned int Dimension, unsigned int Index = 0, class Vector , typename Scalar = typename std::remove_cv<typename std::remove_reference<decltype(std::declval<Vector>()[0])>::type>::type>
static Vector **join** (const Vector &first, const Vector &second)
Join two points.
- template<class Operator , unsigned int Dimension, unsigned int Index = 0, class Vector , typename Scalar = typename std::remove_cv<typename std::remove_reference<decltype(std::declval<Vector>()[0])>::type>::type>
static Vector **apply** (const Vector &first, const Vector &second)
Apply operator to all elements.
- template<unsigned int Dimension, class Vector , typename Scalar >
static constexpr Scalar **radius** (const **magrathea::HyperSphere**< Dimension, Vector, Scalar > &hypersphere)
Radius of an hypersphere.
- template<unsigned int Dimension, class Vector , typename Scalar >
static constexpr Scalar **radius** (const **magrathea::HyperCube**< Dimension, Vector, Scalar > &hypercube)
Radius of an hypercube.
- template<unsigned int Dimension, class Vector , typename Scalar , typename Type = Scalar>
static **magrathea::HyperCube**< Dimension, Vector, Scalar > **cubify** (const **magrathea::HyperSphere**< Dimension, Vector, Scalar > &hypersphere, Type factor=Type(1))
Hypersphere to hypercube conversion.
- template<unsigned int Dimension, class Vector , typename Scalar , typename Type = Scalar>
static **magrathea::HyperSphere**< Dimension, Vector, Scalar > **spherify** (const **magrathea::HyperCube**< Dimension, Vector, Scalar > &hypercube, Type factor=Type(1))
Hypercube to hypersphere conversion.
- template<template< unsigned int, class, typename > class First, template< unsigned int, class, typename > class Second, unsigned int Dimension, class Vector , typename Scalar >
static bool **collide** (const First< Dimension, Vector, Scalar > &first, const Second< Dimension, Vector, Scalar > &second)
Collision between two hyperobjects.
- template<template< unsigned int, class, typename > class Object, unsigned int Dimension, class Vector , typename Scalar , class = typename std::enable_if<Dimension == 3>::type>
static bool **collide** (const Object< Dimension, Vector, Scalar > &object, const **Cone**< Vector, Scalar > &cone, const double &inspheresize)
Collision between an hyperobject and a cone.

Interpolation

- template<typename Type , class Container , class = typename std::enable_if<std::is_convertible<Type, typename std::remove_cv<typename std::remove_reference<decltype(std::declval<Container>()[0])>::type>::type>>::value>::type>
static Type **interpolate** (const Type x0, const Container &x, const Container &y)
Linear interpolation.

- template<typename Type , class Container , class = typename std::enable_if<std::is_convertible<Type, typename std::remove_cv<typename std::remove_reference<decltype(std::declval<Container>())[0]>::type>::type>::value>::type>
static Type **interpolate2** (const Type x0, const Container &x, const Container &y)
Linear interpolation.
- template<typename Type , class Container , class = typename std::enable_if<std::is_convertible<Type, typename std::remove_cv<typename std::remove_reference<decltype(std::declval<Container>())[0]>::type>::type>::value>::type>
static Type **interpolate** (const Type x0, const Container &x, const Container &y, const Container &dydx)
Cubic spline interpolation.
- template<class Container , class... Containers, class = typename std::enable_if<sizeof...(Containers) == 2 || sizeof...(Containers) == 3>::type>
static Container **reinterpolate** (const Container &x0, Containers &&...containers)
Container reinterpolation.
- template<int Direction = 0, typename Type , class Container , typename T = Type, class = typename std::enable_if<std::is_convertible<Type, typename std::remove_cv<typename std::remove_reference<decltype(std::declval<Container>())[0]>::type>::type>::value>::type>
static Type **differentiate** (const Type x0, const Container &x, const Container &y, const unsigned int neighbourhood=1)
Fornberg differentiation.
- template<int Derivative = 0, typename Type , class Container , typename T = Type, class = typename std::enable_if<(Derivative >= 0>
static::type Type **filter** (const Type x0, const Container &x, const Container &y, const unsigned int neighbourhood=1)
Savitzky-Golay filter.

Evolution

- template<class Container , typename Type , class = typename std::enable_if<std::is_same<Type, typename std::remove_cv<typename std::remove_reference<decltype(std::declval<Container>())[0]>::type>::type>::value>::type>
static Container **reverse** (const Container &container, const Type value)
Reverse.
- template<class Container , typename Type = typename std::remove_cv<typename std::remove_reference<decltype(std::declval<Container>())[0]>::type>::type, class Function , class = typename std::enable_if<!std::is_function<typename std::result_of<-Function(Type, Type)>::type>::value>::type>
static Container **smooth** (const Container &x, const Container &y, Function &&kernel, const unsigned int window=0)
Kernel smoother.
- template<class Container , typename Type = typename std::remove_cv<typename std::remove_reference<decltype(std::declval<Container>())[0]>::type>::type, class = typename std::enable_if<std::is_convertible<Type, typename std::remove_cv<typename std::remove_reference<decltype(std::declval<Container>())[0]>::type>::type>::value>::type>
static Container **integrate** (const Container &x, const Container &y, const Type value=Type())
Integration.
- template<int Direction = 0, class Container , typename Type = typename std::remove_cv<typename std::remove_reference<decltype(std::declval<Container>())[0]>::type>::type, class = typename std::enable_if<std::is_convertible<Type, typename std::remove_cv<typename std::remove_reference<decltype(std::declval<Container>())[0]>::type>::type>::value>::type>
static Container **derive** (const Container &x, const Container &y, const unsigned int neighbourhood=1)
Derivative.

Test

- static int **example** ()
Example function.

6.51.1 Detailed Description

List of utilities for raytracing.

Provides a list of general purpose utilities for raytracing like parallelization, collision detection or interpolation.

6.51.2 Member Function Documentation

6.51.2.1 template<class Operator , unsigned int Dimension, unsigned int Index, class Vector , typename Scalar > Vector Utility::apply (const Vector & *first*, const Vector & *second*) [inline], [static]

Apply operator to all elements.

Computes the vector resulting from the application of an operator to all elements of two vectors.

Template Parameters

<i>Operator</i>	Operator type.
<i>Dimension</i>	Number of space dimension.
<i>Index</i>	Inner computation index.
<i>Vector</i>	Position vector type.
<i>Scalar</i>	Scalar data type.

Parameters

in	<i>first</i>	First point.
in	<i>second</i>	Second point.

Returns

Resulting vector.

6.51.2.2 template<template< unsigned int, class, typename > class First, template< unsigned int, class, typename > class Second, unsigned int Dimension, class Vector , typename Scalar > bool Utility::collide (const First< Dimension, Vector, Scalar > & *first*, const Second< Dimension, Vector, Scalar > & *second*) [inline], [static]

Collision between two hyperobjects.

Detects collision between two geometrical objects in arbitrary dimension treated as hyperspheres.

Template Parameters

<i>First</i>	First object type.
<i>Second</i>	Second object type.
<i>Dimension</i>	Number of space dimension.
<i>Vector</i>	Position vector type.
<i>Scalar</i>	Scalar data type.

Parameters

in	<i>first</i>	First object.
in	<i>second</i>	Second object.

Returns

True if collision, false otherwise.

6.51.2.3 template<template< unsigned int, class, typename > class Object, unsigned int Dimension, class Vector, typename Scalar , class > bool Utility::collide (const Object< Dimension, Vector, Scalar > & *object*, const Cone< Vector, Scalar > & *cone*, const double & *insphereSize*) [inline], [static]

Collision between an hyperobject and a cone.

Detects collision between a geometrical object and a three dimensional cone.

Template Parameters

<i>Object</i>	Object type.
<i>Dimension</i>	Number of space dimension.
<i>Vector</i>	Position vector type.
<i>Scalar</i>	Scalar data type.

Parameters

in	<i>object</i>	Geometrical object.
in	<i>cone</i>	Three dimensional cone.

Returns

True if collision, false otherwise.

6.51.2.4 template<unsigned int Dimension, unsigned int Index, class Vector , typename Scalar > Vector Utility::cross (const Vector & *first*, const Vector & *second*) [inline], [static]

Three dimensional cross product.

Computes the cross product of two vectors in three dimensions.

Template Parameters

<i>Dimension</i>	Number of space dimension.
<i>Index</i>	Inner computation index.
<i>Vector</i>	Position vector type.
<i>Scalar</i>	Scalar data type.

Parameters

in	<i>first</i>	First point.
in	<i>second</i>	Second point.

Returns

Result of the cross product.

6.51.2.5 template<unsigned int Dimension, class Vector , typename Scalar , typename Type > magrathea::HyperCube< Dimension, Vector, Scalar > Utility::cubify (const magrathea::HyperSphere< Dimension, Vector, Scalar > & *hypersphere*, Type *scale* = Type(1)) [inline], [static]

Hypersphere to hypercube conversion.

Converts a hypersphere to a hypercubecube with its diagonal equals to the diameter with the provided scaling.

Template Parameters

<i>Dimension</i>	Number of space dimension.
<i>Vector</i>	Position vector type.
<i>Scalar</i>	Scalar data type.

Parameters

in	<i>hypersphere</i>	Input hypersphere.
in	<i>scale</i>	Scaling factor.

Returns

Hypercube resulting from the hypersphere conversion.

6.51.2.6 template<int *Direction*, class *Container* , typename *Type* , class > *Container Utility*::*derive* (const *Container* & *x*, const *Container* & *y*, const unsigned int *neighbourhood* = 1) [static]

Derivative.

Computes the derivative of the ordinates regarding to the abscissae.

Template Parameters

<i>Direction</i>	Sign for centered, backward or forward differentiation.
<i>Container</i>	Container type.
<i>Type</i>	Data type.

Parameters

in	<i>x</i>	Abscissae.
in	<i>y</i>	Ordinates.
in	<i>neighbourhood</i>	Computing distance.

Returns

Derivative of ordinates regarding to the abscissae.

6.51.2.7 template<int *Direction*, typename *Type* , class *Container* , typename *T* , class > *Type Utility*::*differentiate* (const *Type* *x0*, const *Container* & *x*, const *Container* & *y*, const unsigned int *neighbourhood* = 1) [inline], [static]

Fornberg differentiation.

Computes the value of the derivative at the given position using a fourth order Fornberg algorithm. The greater the neighbourhood, the greater the computing distance is.

Template Parameters

<i>Direction</i>	Sign for centered, backward or forward differentiation.
<i>Type</i>	Data type.
<i>Container</i>	Container type.
<i>T</i>	Conversion type.

Parameters

in	<i>x0</i>	Differentiation abscissa.
in	<i>x</i>	Abscissae.
in	<i>y</i>	Ordinates.
in	<i>neighbourhood</i>	Computing distance.

Returns

Derivative at the provided abscissa.

6.51.2.8 template<unsigned int Dimension, unsigned int Index, class Vector , typename Scalar > constexpr Scalar Utility::distance (const Vector & *first*, const Vector & *second*) [static]

Euclidian distance.

Computes the euclidian distance between two points.

Template Parameters

<i>Dimension</i>	Number of space dimension.
<i>Index</i>	Inner computation index.
<i>Vector</i>	Position vector type.
<i>Scalar</i>	Scalar data type.

Parameters

in	<i>first</i>	First point.
in	<i>second</i>	Second point.

Returns

Value of the distance.

6.51.2.9 template<unsigned int Dimension, unsigned int Index, class Vector , typename Scalar > constexpr Scalar Utility::dot (const Vector & *first*, const Vector & *second*) [static]

Dot product.

Computes the dot product of two vectors.

Template Parameters

<i>Dimension</i>	Number of space dimension.
<i>Index</i>	Inner computation index.
<i>Vector</i>	Position vector type.
<i>Scalar</i>	Scalar data type.

Parameters

in	<i>first</i>	First point.
in	<i>second</i>	Second point.

Returns

Value of the dot product.

6.51.2.10 int Utility::example () [static]

Example function.

Tests and demonstrates the use of [Utility](#).

Returns

0 if no error.

6.51.2.11 template<int Derivative, typename Type , class Container , typename T , class > Type Utility::filter (const Type x_0 , const Container & x , const Container & y , const unsigned int $neighbourhood = 1$) [inline], [static]

Savitzky-Golay filter.

Computes the smoothed value of the n-th derivative at the given position using a fourth order Savitzky-Golay algorithm. The greater the neighbourhood, the greater the computing distance is.

Template Parameters

<i>Derivative</i>	Order of the derivative.
<i>Type</i>	Data type.
<i>Container</i>	Container type.
<i>T</i>	Conversion type.

Parameters

in	x_0	Differentiation abscissa.
in	x	Abscissae.
in	y	Ordinates.
in	<i>neighbourhood</i>	Computing distance.

Returns

Filtered derivative at the provided abscissa.

6.51.2.12 template<class Container , typename Type , class > Container Utility::integrate (const Container & x , const Container & y , const Type $value = Type()$) [static]

Integration.

Computes the integral of the ordinates regarding to the abscissae and starting from the specified value.

Template Parameters

<i>Container</i>	Container type.
<i>Type</i>	Data type.

Parameters

in	x	Abscissae.
in	y	Ordinates.

Returns

Integral of ordinates regarding to the abscissae.

6.51.2.13 template<typename Type , class Container , class > Type Utility::interpolate (const Type x_0 , const Container & x , const Container & y) [inline], [static]

Linear interpolation.

Interpolates the value at the given position linearly.

Template Parameters

<i>Type</i>	Data type.
<i>Container</i>	Container type.

Parameters

in	<i>x0</i>	Interpolation abscissa.
in	<i>x</i>	Abscissae.
in	<i>y</i>	Ordinates.

Returns

Interpolated ordinate.

6.51.2.14 template<typename Type , class Container , class > Type Utility::interpolate (const Type *x0*, const Container & *x*, const Container & *y*, const Container & *dydx*) [inline], [static]

Cubic spline interpolation.

Interpolates the value at the given position using a cubic spline with the specified derivative.

Template Parameters

<i>Type</i>	Data type.
<i>Container</i>	Container type.

Parameters

in	<i>x0</i>	Interpolation abscissa.
in	<i>x</i>	Abscissae.
in	<i>y</i>	Ordinates.
in	<i>dydx</i>	Derivatives.

Returns

Interpolated ordinate.

6.51.2.15 template<typename Type , class Container , class > Type Utility::interpolate2 (const Type *x0*, const Container & *x*, const Container & *y*) [inline], [static]

Linear interpolation.

Interpolates the value at the given position linearly, with decreasing x.

Template Parameters

<i>Type</i>	Data type.
<i>Container</i>	Container type.

Parameters

in	<i>x0</i>	Interpolation abscissa.
in	<i>x</i>	Abscissae.

in	<i>y</i>	Ordinates.
----	----------	------------

Returns

Interpolated ordinate.

6.51.2.16 template<class Type > std::array< std::array< Type, 2 >, 2 > Utility::invMatrix2d (const std::array< std::array< Type, 2 >, 2 > & *A*) [inline], [static]

Inverse matrix.

Inverse 2D matrix.

Template Parameters

Type	type of matrix components.
------	----------------------------

Parameters

in	<i>A</i>	array 2D Matrix. return Array 2D inverted matrix
----	----------	--

6.51.2.17 template<class Type > std::array< std::array< Type, 3 >, 3 > Utility::invMatrix3d (const std::array< std::array< Type, 3 >, 3 > & *A*) [inline], [static]

Inverse matrix.

Inverse 3D matrix.

Template Parameters

Type	type of matrix components.
------	----------------------------

Parameters

in	<i>A</i>	array 3D Matrix. return Array 3D inverted matrix
----	----------	--

6.51.2.18 template<unsigned int Dimension, unsigned int Index, class Vector , typename Scalar > Vector Utility::join (const Vector & *first*, const Vector & *second*) [inline], [static]

Join two points.

Computes the vector resulting from going from the first point to the second point.

Template Parameters

<i>Dimension</i>	Number of space dimension.
<i>Index</i>	Inner computation index.
<i>Vector</i>	Position vector type.
<i>Scalar</i>	Scalar data type.

Parameters

in	<i>first</i>	First point.
in	<i>second</i>	Second point.

Returns

Resulting joining vector.

```
6.51.2.19 template<int Default, typename Type , class Function , class > double Utility::parallelize ( const Type nsteps,  
Function && function, const int nthreads = (Default != 0) ? (Default) : (std::thread::hardware_concurrency()) ) [static]
```

Parallelize a loop.

Executes the provided function on each index of the loop using the specified number of threads.

Template Parameters

<i>Default</i>	Default concurrency where zero means hardware concurrency.
<i>Type</i>	Loop index type.
<i>Function</i>	Function type taking a loop index as argument.

Parameters

in	<i>nsteps</i>	Total number of steps.
in	<i>function</i>	Function.
in	<i>nthreads</i>	Number of threads.

Returns

Elapsed time in seconds.

```
6.51.2.20 template<int Default, typename Type , class Function , class > double Utility::parallelize ( const  
Type & first, const Type & last, const Type & increment, Function && function, const int nthreads =  
(Default != 0) ? (Default) : (std::thread::hardware_concurrency()) ) [static]
```

Parallelize an iteration over a range of values.

Executes the provided function on each values of the range using the provided increment and the specified number of threads.

Template Parameters

<i>Default</i>	Default concurrency where zero means hardware concurrency.
<i>Type</i>	Value type.
<i>Function</i>	Function type taking a value as argument.

Parameters

in	<i>first</i>	First value.
in	<i>last</i>	Last value.
in	<i>increment</i>	Increment.
in	<i>function</i>	Function.
in	<i>nthreads</i>	Number of threads.

Returns

Elapsed time in seconds.

```
6.51.2.21 template<int Default, typename Iterator , class Function , class > double Utility::parallelize
( const Iterator & first, const Iterator & last, Function && function, const int nthreads =
(Default != 0) ? (Default) : (std::hardware_concurrency()) ) [static]
```

Parallelize an iteration over a range of iterators.

Executes the provided function on each element of the iterator range using the specified number of threads.

Template Parameters

<i>Default</i>	Default concurrency where zero means hardware concurrency.
<i>Iterator</i>	Iterator type.
<i>Function</i>	Function type taking an element as argument.

Parameters

in	<i>first</i>	First iterator.
in	<i>last</i>	Last iterator.
in	<i>function</i>	Function.
in	<i>nthreads</i>	Number of threads.

Returns

Elapsed time in seconds.

```
6.51.2.22 template<unsigned int Dimension, class Vector , typename Scalar > constexpr Scalar Utility::radius ( const
magrathea::HyperSphere< Dimension, Vector, Scalar > & hypersphere ) [static]
```

Radius of an hypersphere.

Computes the radius of the provided hypersphere.

Template Parameters

<i>Dimension</i>	Number of space dimension.
<i>Vector</i>	Position vector type.
<i>Scalar</i>	Scalar data type.

Parameters

in	<i>hypersphere</i>	Hypersphere.
----	--------------------	--------------

Returns

Value of the radius.

```
6.51.2.23 template<unsigned int Dimension, class Vector , typename Scalar > constexpr Scalar Utility::radius ( const
magrathea::HyperCube< Dimension, Vector, Scalar > & hypercube ) [static]
```

Radius of an hypercube.

Computes the radius corresponding to half the diagonal of the provided hypercube.

Template Parameters

<i>Dimension</i>	Number of space dimension.
<i>Vector</i>	Position vector type.
<i>Scalar</i>	Scalar data type.

Parameters

in	<i>hypercube</i>	Hypercube.
----	------------------	------------

Returns

Value of the radius.

6.51.2.24 template<class Container , class... Containers, class > Container Utility::reinterpolate (const Container & *x0*, Containers &&... *containers*) [inline], [static]

Container reinterpolation.

Interpolates each value of the container.

Template Parameters

<i>Container</i>	Container type.
<i>Containers</i>	Containers types.

Parameters

in	<i>x0</i>	Interpolation abscissae.
in	<i>containers</i>	Input containers.

Returns

Interpolated ordinates.

6.51.2.25 template<class Container , typename Type , class > Container Utility::reverse (const Container & *container*, const Type *value*) [static]

Reverse.

Reverses a container by subtracting a value.

Template Parameters

<i>Container</i>	Container type.
<i>Type</i>	Data type.

Parameters

in	<i>container</i>	Container.
in	<i>value</i>	Value to be subtracted.

Returns

Reversed vector.

6.51.2.26 template<class Container , typename Type , class Function , class > Container Utility::smooth (const Container & x, const Container & y, Function && kernel, const unsigned int window = 0) [static]

Kernel smoother.

Smoothes a serie of data using a kernel smoother and a window function on the first neighbours.

Template Parameters

<i>Container</i>	Container type.
<i>Type</i>	Data type.
<i>Function</i>	Function type taking two data as argument.

Parameters

in	x	Abscissae.
in	y	Ordinates.
in	kernel	Kernel function.
in	window	Cut on the provided number of neighbours.

Returns

Smoothed ordinates.

6.51.2.27 template<unsigned int Dimension, class Vector , typename Scalar , typename Type >
magrathea::HyperSphere< Dimension, Vector, Scalar > Utility::spherify (const **magrathea::HyperCube**< Dimension, Vector, Scalar > & hypercube, Type scale = Type(1)) [inline], [static]

Hypercube to hypersphere conversion.

Converts a hypercube to a hypersphere with its diameter equals to the diagonal with the provided scaling.

Template Parameters

<i>Dimension</i>	Number of space dimension.
<i>Vector</i>	Position vector type.
<i>Scalar</i>	Scalar data type.

Parameters

in	hypercube	Input hypercube.
in	scale	Scaling factor.

Returns

Hypersphere resulting from the hypercube conversion.

The documentation for this exception was generated from the following file:

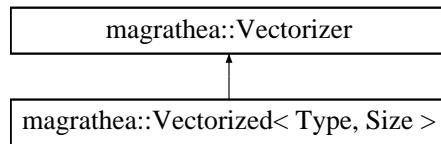
- /data/home/mbreton/magrathea_pathfinder/src/utility.h

6.52 **magrathea::Vectorized**< Type, Size > Exception Template Reference

Basic vectorized container.

```
#include <vectorized.h>
```

Inheritance diagram for magrathea::Vectorized< Type, Size >:



Public Member Functions

Lifecycle

- **Vectorized ()**
Implicit empty constructor.
- template<typename FundamentalType = Type, class = typename std::enable_if<std::is_fundamental<FundamentalType>::value>::type>
Vectorized (const **Vectorized**< FundamentalType, Size > &source)
Implicit conversion constructor.
- template<typename OtherType = Type, class... Misc, class = typename std::enable_if<std::is_convertible<OtherType, Type>::value>::type>
Vectorized (const std::initializer_list< OtherType > &source, const Misc &...misc)
Implicit initializer list constructor.
- template<class... Misc, class = typename std::enable_if<sizeof...(Misc) != 0>::type>
Vectorized (const Misc &...misc)
Explicit generic constructor.

Vectorization

- Type & **operator[]** (const unsigned int i)
Direct access to the element.
- const Type & **operator[]** (const unsigned int i) const
Immutable direct access to the element.
- **Vectorized**< Type, Size > & **resize** (const unsigned int n)
Resize the container.
- **Vectorized**< Type, Size > & **reserve** (const unsigned int n)

Static Public Member Functions

Static vectorization

- static constexpr unsigned int **size** ()
Get the size of the container.
- static constexpr unsigned int **capacity** ()
- static constexpr bool **constant** ()
Get whether the container has a constant size.
- static constexpr bool **boolean** ()
Get whether the container has a boolean type.
- static constexpr std::array< unsigned int, 1 > **parameters** ()
Get the template parameters.
- static Type **type** ()
Get the data type.

Test

- static int **example** ()
Example function.

Protected Attributes

Data members

- Type `_data` [Size]
Data contents.

Friends

Stream

- template<typename SelfType , unsigned int SelfSize>
`std::ostream & operator<< (std::ostream &lhs, const Vectorized< SelfType, SelfSize > &rhs)`
Output stream operator.

Additional Inherited Members

6.52.1 Detailed Description

`template<typename Type = double, unsigned int Size = 1>exception magrathea::Vectorized< Type, Size >`

Basic vectorized container.

This class is the direct derivation of [Vectorizer](#). It provides the most basic vectorized container without adding new functionalities to the abstract class.

Template Parameters

<code>Type</code>	Data type.
<code>Size</code>	Number of elements.

6.52.2 Constructor & Destructor Documentation

6.52.2.1 `template<typename Type , unsigned int Size> magrathea::Vectorized< Type, Size >::Vectorized()` [inline]

Implicit empty constructor.

Does nothing.

6.52.2.2 `template<typename Type , unsigned int Size> template<typename FundamentalType , class > magrathea::Vectorized< Type, Size >::Vectorized(const Vectorized< FundamentalType, Size > & source)` [inline]

Implicit conversion constructor.

Provides an implicit conversion from a fundamental type contents.

Template Parameters

<code>FundamentalType</code>	(Fundamental data type.)
------------------------------	--------------------------

Parameters

<code>in</code>	<code>source</code>	Source of the copy.
-----------------	---------------------	---------------------

6.52.2.3 template<typename Type , unsigned int Size> template<typename OtherType , class... Misc, class > magrathea::Vectorized< Type, Size >::Vectorized (const std::initializer_list< OtherType > & source, const Misc &... misc) [inline]

Implicit initializer list constructor.

Provides an implicit conversion from an initializer list.

Template Parameters

<i>OtherType</i>	(Other data type.)
<i>Misc</i>	(Miscellaneous types.)

Parameters

in	<i>source</i>	Source of the copy.
in	<i>misc</i>	Miscellaneous arguments.

6.52.2.4 template<typename Type , unsigned int Size> template<class... Misc, class > magrathea::Vectorized< Type, Size >::Vectorized (const Misc &... misc) [inline], [explicit]

Explicit generic constructor.

Provides a generic interface to all constructors of the base class. Before calling the associated constructor of the base class, the contents is initialized.

Template Parameters

<i>Misc</i>	(Miscellaneous types.)
-------------	------------------------

Parameters

in	<i>misc</i>	Miscellaneous arguments.
----	-------------	--------------------------

6.52.3 Member Function Documentation

6.52.3.1 template<typename Type , unsigned int Size> constexpr bool magrathea::Vectorized< Type, Size >::boolean () [static]

Get whether the container has a boolean type.

Returns true if the container has a boolean type, false otherwise. This function is required by the vectorization mechanism.

Returns

Copy of true if the container has a boolean type.

6.52.3.2 template<typename Type = double, unsigned int Size = 1> static constexpr unsigned int magrathea::Vectorized< Type, Size >::capacity () [static]

6.52.3.3 template<typename Type , unsigned int Size> constexpr bool magrathea::Vectorized< Type, Size >::constant () [static]

Get whether the container has a constant size.

Returns true if the container has a constant size, false otherwise. This function is required by the vectorization mechanism.

Returns

Copy of true.

6.52.3.4 template<typename Type , unsigned int Size> int magrathea::Vectorized< Type, Size >::example () [static]

Example function.

Tests and demonstrates the use of [Vectorized](#).

Returns

0 if no error.

6.52.3.5 template<typename Type , unsigned int Size> Type & magrathea::Vectorized< Type, Size >::operator[] (const unsigned int i) [inline]

Direct access to the element.

Provides a direct access to the specified element. This function is required by the vectorization mechanism.

Parameters

in	i	Index of the element.
----	---	-----------------------

Returns

Reference to the element.

6.52.3.6 template<typename Type , unsigned int Size> const Type & magrathea::Vectorized< Type, Size >::operator[] (const unsigned int i) const [inline]

Immutable direct access to the element.

Provides a constant direct access to the specified element. This function is required by the vectorization mechanism.

Parameters

in	i	Index of the element.
----	---	-----------------------

Returns

Const reference to the element.

6.52.3.7 template<typename Type , unsigned int Size> constexpr std::array< unsigned int, 1 > magrathea::Vectorized< Type, Size >::parameters () [static]

Get the template parameters.

Returns an array containing the template parameters. This function is required by the vectorization mechanism.

Returns

Copy of an array of parameters.

6.52.3.8 template<typename Type = double, unsigned int Size = 1> Vectorized<Type, Size>& magrathea::Vectorized<Type, Size >::reserve (const unsigned int n) [inline]

6.52.3.9 template<typename Type , unsigned int Size> Vectorized< Type, Size > & magrathea::Vectorized< Type, Size >::resize (const unsigned int n) [inline]

Resize the container.

Resizes the container and returns a reference to it. This function is required by the vectorization mechanism.

Parameters

in	n	New size.
----	---	-----------

Returns

Self reference.

Exceptions

<i>std::length_error</i>	The container cannot be resized.
--------------------------	----------------------------------

6.52.3.10 template<typename Type , unsigned int Size> constexpr unsigned int magrathea::Vectorized< Type, Size >::size () [static]

Get the size of the container.

Returns the current number of elements. This function is required by the vectorization mechanism.

Returns

Copy of the size.

6.52.3.11 template<typename Type , unsigned int Size> Type magrathea::Vectorized< Type, Size >::type () [inline], [static]

Get the data type.

Returns a copy of the default value of the data type.

Returns

Copy of the default value of the data type.

6.52.4 Friends And Related Function Documentation

6.52.4.1 template<typename Type = double, unsigned int Size = 1> template<typename SelfType , unsigned int SelfSize> std::ostream& operator<< (std::ostream & lhs, const Vectorized< SelfType, SelfSize > & rhs) [friend]

[Output](#) stream operator.

Adds each element to the stream using the fill character to separate the elements.

Template Parameters

<i>SelfType</i>	Data type.
<i>SelfSize</i>	Number of elements.

Parameters

<i>in, out</i>	<i>lhs</i>	Left-hand side stream.
<i>in</i>	<i>rhs</i>	Right-hand side container.

Returns

[Output](#) stream.

6.52.5 Member Data Documentation

6.52.5.1 template<typename Type = double, unsigned int Size = 1> Type magrathea::Vectorized< Type, Size >::data[Size] [protected]

Data contents.

The documentation for this exception was generated from the following file:

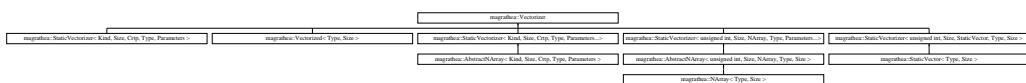
- /data/home/mbreton/magrathea_pathfinder/src/magrathea/[vectorized.h](#)

6.53 magrathea::Vectorizer Class Reference

Helper base class for generic vectorization.

```
#include <vectorizer.h>
```

Inheritance diagram for magrathea::Vectorizer:



Public Member Functions

Vectorization

- [Vectorizer & operator\[\]](#) (const unsigned int i)
Direct access to the element.
- const [Vectorizer & operator\[\]](#) (const unsigned int i) const
Immutable direct access to the element.
- [Vectorizer & resize](#) (const unsigned int n)
Resize the container.
- unsigned int [size](#) () const
Get the size of the container.
- bool [constant](#) () const
Get whether the container has a constant size.
- bool [boolean](#) () const
Get whether the container has a boolean type.
- std::array< unsigned int, 0 > [parameters](#) () const
Get the template parameters.
- void [type](#) () const
Get the data type.

Static Public Member Functions

Check

- template<bool Exception = true, class FirstType , class SecondType , class = typename std::enable_if<(!std::is_base_of<Vectorizer, FirstType>::value) || (!std::is_base_of<Vectorizer, SecondType>::value)>::type>
static constexpr bool **check** (const FirstType &, const SecondType &)
Check compatibility with at least one non-vectorized type.
- template<bool Exception = true, class FirstType , class SecondType , class = typename std::enable_if<(std::is_base_of<-Vectorizer, FirstType>::value) && (std::is_base_of<Vectorizer, SecondType>::value)>::type, class = typename std::enable_if<(-FirstType::constant()) && (SecondType::constant())>::type>
static constexpr bool **check** (const FirstType &, const SecondType &)
- template<bool Exception = true, class FirstType , class SecondType , class = typename std::enable_if<(std::is_base_of<Vectorizer, FirstType>::value) && (std::is_base_of<Vectorizer, SecondType>::value)>::type, class = typename std::enable_if<(!FirstType::constant()) || (!SecondType::constant())>::type, class = void>
static bool **check** (const FirstType &first, const SecondType &second)

Getters

- template<typename IntegralType , IntegralType Value, typename DummyType = unsigned int, class = typename std::enable_if<(std::is_integral<IntegralType>::value) && (std::is_convertible<DummyType, unsigned int>::value)>::type>
static constexpr IntegralType **get** (const std::integral_constant< IntegralType, Value >, const DummyType=DummyType())
Integral constant getter.
- template<typename DataType , typename DummyType = unsigned int, class = typename std::enable_if<(!std::is_base_of<-Vectorizer, DataType>::value) && (std::is_convertible<DummyType, unsigned int>::value)>::type>
static DataType & **get** (DataType &source, const DummyType=DummyType())
Non-vectorized getter.
- template<typename DataType , typename DummyType = unsigned int, class = typename std::enable_if<(!std::is_base_of<-Vectorizer, DataType>::value) && (std::is_convertible<DummyType, unsigned int>::value)>::type>
static const DataType & **get** (const DataType &source, const DummyType=DummyType())
Immutable non-vectorized getter.
- template<class VectorizedType , class = typename std::enable_if<std::is_base_of<Vectorizer, VectorizedType>::value>::type, typename DataType = typename std::remove_reference<decltype(VectorizedType::type())>::type>
static DataType & **get** (VectorizedType &source, const unsigned int i)
Vector element getter.
- template<class VectorizedType , class = typename std::enable_if<std::is_base_of<Vectorizer, VectorizedType>::value>::type, typename DataType = typename std::remove_reference<decltype(VectorizedType::type())>::type>
static const DataType & **get** (const VectorizedType &source, const unsigned int i)
Immutable vector element getter.

Setters

- template<class VectorizedType , class = typename std::enable_if<std::is_base_of<Vectorizer, VectorizedType>::value>::type>
static VectorizedType & **set** (VectorizedType &destination)
Empty setter.
- template<class VectorizedType , class = typename std::enable_if<std::is_base_of<Vectorizer, VectorizedType>::value>::type, class GenericType >
static VectorizedType & **set** (VectorizedType &destination, const GenericType &source)
Copy setter.
- template<class VectorizedType , class = typename std::enable_if<std::is_base_of<Vectorizer, VectorizedType>::value>::type, class GenericType , class First , class Second , class... Others, class = typename std::enable_if<((sizeof...(Others) != 0) && (sizeof...(-Others) != 1))>::type>
static VectorizedType & **set** (VectorizedType &destination, const GenericType &source, const First &first, const Second &second, const Others &...others)
Variadic setter.
- template<class VectorizedType , class = typename std::enable_if<std::is_base_of<Vectorizer, VectorizedType>::value>::type, class GenericType , typename SizeType = std::true_type, class = typename std::enable_if<(std::is_same<SizeType, std::true_type>::value) || (std::is_convertible<SizeType, unsigned int>::value)>::type>
static VectorizedType & **set** (VectorizedType &destination, const GenericType &source, const unsigned int pos, const SizeType num=SizeType())

Partial setter.

- template<class VectorizedType , class = typename std::enable_if<std::is_base_of<Vectorizer, VectorizedType>::value>::type, typename DataType = typename std::remove_reference<decltype(VectorizedType::type())>::type, typename SizeType = std::true_type, class = typename std::enable_if<(std::is_convertible<DataType, typename std::remove_reference<decltype(std::declval<VectorizedType>()[0])>::type>::value) && ((std::is_same<SizeType, std::true_type>::value) || (std::is_convertible<SizeType, unsigned int>::value))>::type>
 static VectorizedType & **set** (VectorizedType &destination, const std::initializer_list< DataType > &source, const unsigned int pos=0, const SizeType num=SizeType())

Partial list setter.

- template<class VectorizedType , class = typename std::enable_if<std::is_base_of<Vectorizer, VectorizedType>::value>::type, typename IteratorType , typename SizeType = std::true_type, class = typename std::enable_if<(std::is_same<SizeType, std::true_type>::value) || (std::is_convertible<SizeType, unsigned int>::value)>::type>
 static VectorizedType & **set** (VectorizedType &destination, const IteratorType &first, const IteratorType &last, const unsigned int pos=0, const SizeType num=SizeType(), typename std::iterator_traits< IteratorType >::iterator_category *=nullptr)

Partial range setter.

- template<class VectorizedType , class = typename std::enable_if<std::is_base_of<Vectorizer, VectorizedType>::value>::type, class GenericType , class MaskType , typename DummyType = unsigned int, class = typename std::enable_if<(std::is_base_of<Vectorizer, MaskType>::value) && (std::is_convertible<DummyType, unsigned int>::value)>::type>
 static VectorizedType & **set** (VectorizedType &destination, const GenericType &source, const MaskType &mask, const DummyType=DummyType())

Masked setter.

- template<class VectorizedType , class = typename std::enable_if<std::is_base_of<Vectorizer, VectorizedType>::value>::type, typename DataType = typename std::remove_reference<decltype(VectorizedType::type())>::type, class MaskType , typename DummyType = unsigned int, class = typename std::enable_if<(std::is_base_of<Vectorizer, MaskType>::value) && (std::is_convertible<DummyType, unsigned int>::value)>::type>
 static VectorizedType & **set** (VectorizedType &destination, const std::initializer_list< DataType > &source, const MaskType &mask, const DummyType=DummyType())

Masked list setter.

- template<class VectorizedType , class = typename std::enable_if<std::is_base_of<Vectorizer, VectorizedType>::value>::type, typename IteratorType , class MaskType , typename DummyType = unsigned int, class = typename std::enable_if<(std::is_base_of<Vectorizer, MaskType>::value) && (std::is_convertible<DummyType, unsigned int>::value)>::type>
 static VectorizedType & **set** (VectorizedType &destination, const IteratorType &first, const IteratorType &last, const MaskType &mask, const DummyType=DummyType(), typename std::iterator_traits< IteratorType >::iterator_category *=nullptr)

*Masked range setter.***Test**

- static int **example** ()

*Example function.***Protected Member Functions****Protected lifecycle**

- ~Vectorizer** ()

*Protected destructor.***6.53.1 Detailed Description**

Helper base class for generic vectorization.

Provides global functions for vectorization implementation. **Vectorizer** helpers (like **StaticVectorizer** or **DynamicVectorizer**) are derived from this class and have to implement the following functions required by CRTP :

- operator []**

- `resize()`
- `size()`
- `constant()`
- `boolean()`
- `parameters()`
- `type()`

6.53.2 Constructor & Destructor Documentation

6.53.2.1 `magrathea::Vectorizer::~Vectorizer() [inline], [protected], [default]`

Protected destructor.

Does nothing.

6.53.3 Member Function Documentation

6.53.3.1 `bool magrathea::Vectorizer::boolean() const [inline]`

Get whether the container has a boolean type.

Returns true if the container has a boolean type size, false otherwise. This function is required by the vectorization mechanism.

Returns

Copy of true if the container has a boolean type.

Exceptions

<code>std::logic_error</code>	This function should be overloaded by derived classes.
-------------------------------	--

6.53.3.2 `template<bool Exception, class FirstType , class SecondType , class , class , class > bool magrathea::Vectorizer::check(const FirstType & first, const SecondType & second) [inline], [static]`

Check compatibility with at least one non-vectorized type.

Check compatibility between two vectorized types.

Check compatibility between two static vectorized types.

Checks whether the two provided vectorizer have compatible properties.

Template Parameters

<code>Exception</code>	Throw exception or assertion on error.
<code>FirstType</code>	(First type.)
<code>SecondType</code>	(Second type.)

Returns

True whether the two types are compatible, false otherwise.

Checks whether the two provided vectorizer have compatible properties.

Template Parameters

<i>Exception</i>	Throw exception or assertion on error.
<i>FirstType</i>	(First type.)
<i>SecondType</i>	(Second type.)

Parameters

in	<i>first</i>	First argument.
in	<i>second</i>	Second argument.

Returns

True whether the two types are compatible, false otherwise.

Exceptions

<i>std::length_error</i>	Vectorizer sizes are not equal.
--------------------------	---

```
6.53.3.3 template<bool Exception = true, class FirstType , class SecondType , class = typename std::enable_if<(std::is_base_of<Vectorizer, FirstType>::value) && (std::is_base_of<Vectorizer, SecondType>::value)>::type, class = typename std::enable_if<(FirstType::constant()) && (SecondType::constant())>::type> static constexpr bool magrathea::Vectorizer::check ( const FirstType &, const SecondType & ) [static]
```

```
6.53.3.4 template<bool Exception = true, class FirstType , class SecondType , class = typename std::enable_if<(std::is_base_of<Vectorizer, FirstType>::value) && (std::is_base_of<Vectorizer, SecondType>::value)>::type, class = typename std::enable_if<(!FirstType::constant()) || (!SecondType::constant())>::type, class = void> static bool magrathea::Vectorizer::check ( const FirstType & first, const SecondType & second ) [inline], [static]
```

```
6.53.3.5 bool magrathea::Vectorizer::constant ( ) const [inline]
```

Get whether the container has a constant size.

Returns true if the container has a constant size, false otherwise. This function is required by the vectorization mechanism.

Returns

Copy of true if the container has a constant size.

Exceptions

<i>std::logic_error</i>	This function should be overloaded by derived classes.
-------------------------	--

```
6.53.3.6 int magrathea::Vectorizer::example ( ) [static]
```

Example function.

Tests and demonstrates the use of [Vectorizer](#).

Returns

0 if no error.

```
6.53.3.7 template<typename IntegralType , IntegralType Value, typename DummyType , class > constexpr IntegralType
magrathea::Vectorizer::get ( const std::integral_constant< IntegralType, Value > , const DummyType =
DummyType () ) [static]
```

Integral constant getter.

Returns the value of the provided integral constant.

Template Parameters

<i>IntegralType</i>	(Integral type.)
<i>DummyType</i>	(Dummy parameter type.)

Returns

Value of the integral constant.

```
6.53.3.8 template<typename DataType , typename DummyType, class > DataType & magrathea::Vectorizer::get ( DataType &
source, const DummyType = DummyType () ) [inline], [static]
```

Non-vectorized getter.

Returns a reference to the provided non-vectorized source.

Template Parameters

<i>DataType</i>	(Data type.)
<i>DummyType</i>	(Dummy parameter type.)

Parameters

in	<i>source</i>	Argument to get.
----	---------------	------------------

Returns

Reference to the argument.

```
6.53.3.9 template<typename DataType , typename DummyType, class > const DataType & magrathea::Vectorizer::get ( const
DataType & source, const DummyType = DummyType () ) [inline], [static]
```

Immutable non-vectorized getter.

Returns a constant reference to the provided non-vectorized source.

Template Parameters

<i>DataType</i>	(Data type.)
<i>DummyType</i>	(Dummy parameter type.)

Parameters

in	<i>source</i>	Argument to get.
----	---------------	------------------

Returns

Const reference to the argument.

6.53.3.10 template<class VectorizedType , class , typename DataType > DataType & magrathea::Vectorizer::get (VectorizedType & source, const unsigned int *i*) [inline], [static]

Vector element getter.

Returns a reference to the *i*-th element of the provided vectorized source.

Template Parameters

<i>VectorizedType</i>	(Vectorized type.)
<i>DataType</i>	(Data type.)

Parameters

in	<i>source</i>	Source container.
in	<i>i</i>	Index of the element.

Returns

Reference to the specified element.

6.53.3.11 template<class VectorizedType , class , typename DataType > const DataType & magrathea::Vectorizer::get (const VectorizedType & source, const unsigned int *i*) [inline], [static]

Immutable vector element getter.

Returns a constant reference to the *i*-th element of the provided vectorized source.

Template Parameters

<i>VectorizedType</i>	(Vectorized type.)
<i>DataType</i>	(Data type.)

Parameters

in	<i>source</i>	Source container.
in	<i>i</i>	Index of the element.

Returns

Const reference to the specified element.

6.53.3.12 Vectorizer & magrathea::Vectorizer::operator[] (const unsigned int *i*) [inline]

Direct access to the element.

Provides a direct access to the specified element. This function is required by the vectorization mechanism.

Parameters

in	<i>i</i>	Index of the element.
----	----------	-----------------------

Returns

Reference to the element.

Exceptions

<i>std::logic_error</i>	This function should be overloaded by derived classes.
-------------------------	--

6.53.3.13 const Vectorizer & magrathea::Vectorizer::operator[] (const unsigned int *i*) const [inline]

Immutable direct access to the element.

Provides a constant direct access to the specified element. This function is required by the vectorization mechanism.

Parameters

in	<i>i</i>	Index of the element.
----	----------	-----------------------

Returns

Const reference to the element.

Exceptions

<i>std::logic_error</i>	This function should be overloaded by derived classes.
-------------------------	--

6.53.3.14 std::array< unsigned int, 0 > magrathea::Vectorizer::parameters () const [inline]

Get the template parameters.

Returns an array containing the template parameters. This function is required by the vectorization mechanism.

Returns

Copy of an array of parameters.

Exceptions

<i>std::logic_error</i>	This function should be overloaded by derived classes.
-------------------------	--

6.53.3.15 Vectorizer & magrathea::Vectorizer::resize (const unsigned int *n*) [inline]

Resize the container.

Resizes the container and returns a reference to it. This function is required by the vectorization mechanism.

Parameters

in	<i>n</i>	New size.
----	----------	-----------

Returns

Self reference.

Exceptions

<i>std::logic_error</i>	This function should be overloaded by derived classes.
-------------------------	--

6.53.3.16 template<class **VectorizedType** , class > **VectorizedType** & magrathea::Vectorizer::set (**VectorizedType** & *destination*) [inline], [static]

Empty setter.

Does nothing.

Template Parameters

<i>VectorizedType</i>	(Vectorized type.)
-----------------------	-------------------------------------

Parameters

<i>in, out</i>	<i>destination</i>	Destination of the copy.
----------------	--------------------	--------------------------

Returns

Reference to the destination.

6.53.3.17 template<class **VectorizedType** , class , class **GenericType** > **VectorizedType** & magrathea::Vectorizer::set (**VectorizedType** & *destination*, const **GenericType** & *source*) [inline], [static]

Copy setter.

Copies the whole contents of the source to the destination.

Template Parameters

<i>VectorizedType</i>	(Vectorized type.)
<i>GenericType</i>	(Generic type.)

Parameters

<i>in, out</i>	<i>destination</i>	Destination of the copy.
<i>in</i>	<i>source</i>	Source of the copy.

Returns

Reference to the destination.

6.53.3.18 template<class **VectorizedType** , class , class **GenericType** , class **First** , class **Second** , class... **Others**, class > **VectorizedType** & magrathea::Vectorizer::set (**VectorizedType** & *destination*, const **GenericType** & *source*, const **First** & *first*, const **Second** & *second*, const **Others** &... *others*) [inline], [static]

Variadic setter.

Calls recursively the setters for a long list of arguments.

Template Parameters

<i>VectorizedType</i>	(Vectorized type.)
<i>GenericType</i>	(Generic type.)
<i>First</i>	(First type.)
<i>Second</i>	(Second type.)
<i>Others</i>	(Other types.)

Parameters

<i>in,out</i>	<i>destination</i>	Destination of the copy.
<i>in</i>	<i>source</i>	Source of the copy.
<i>in</i>	<i>first</i>	First extra argument.
<i>in</i>	<i>second</i>	Second extra argument.
<i>in</i>	<i>others</i>	Other arguments.

Returns

Reference to the destination.

6.53.3.19 template<class VectorizedType , class , class GenericType , typename SizeType , class > VectorizedType & magrathea::Vectorizer::set (VectorizedType & *destination*, const GenericType & *source*, const unsigned int *pos*, const SizeType *num* = SizeType ()) [inline], [static]

Partial setter.

Copies the contents of the source to a part of the destination.

Template Parameters

<i>VectorizedType</i>	(Vectorized type.)
<i>GenericType</i>	(Generic type.)
<i>SizeType</i>	(Size type.)

Parameters

<i>in,out</i>	<i>destination</i>	Destination of the copy.
<i>in</i>	<i>source</i>	Source of the copy.
<i>in</i>	<i>pos</i>	Starting position of the copy.
<i>in</i>	<i>num</i>	Number of elements to copy.

Returns

Reference to the destination.

6.53.3.20 template<class VectorizedType , class , typename DataType , typename SizeType , class > VectorizedType & magrathea::Vectorizer::set (VectorizedType & *destination*, const std::initializer_list< DataType > & *source*, const unsigned int *pos* = 0, const SizeType *num* = SizeType ()) [inline], [static]

Partial list setter.

Copies the contents of the source to a part of the destination. The first element of the list is copied at the provided position, and the next elements are copied after it. If the list is too small, empty values are added to its end.

Template Parameters

<i>VectorizedType</i>	(Vectorized type.)
<i>DataType</i>	(Data type.)
<i>SizeType</i>	(Size type.)

Parameters

<i>in,out</i>	<i>destination</i>	Destination of the copy.
<i>in</i>	<i>source</i>	Source of the copy.

in	<i>pos</i>	Starting position of the copy.
in	<i>num</i>	Number of elements to copy.

Returns

Reference to the destination.

```
6.53.3.21 template<class VectorizedType , class , typename IteratorType , typename SizeType , class > VectorizedType &
magrathea::Vectorizer::set ( VectorizedType & destination, const IteratorType & first, const IteratorType & last,
const unsigned int pos = 0, const SizeType num = SizeType (), typename std::iterator_traits< IteratorType
>::iterator_category * = nullptr ) [inline], [static]
```

Partial range setter.

Copies the values from the range to a part of the destination. The first element of the range is copied at the provided position, and the next elements are copied after it. The copy stops as soon as the end of the range is encountered or if the number of elements to copy is reached.

Template Parameters

<i>VectorizedType</i>	(Vectorized type.)
<i>IteratorType</i>	(Iterator or pointer type.)
<i>SizeType</i>	(Size type.)

Parameters

in, out	<i>destination</i>	Destination of the copy.
in	<i>first</i>	Iterator to the beginning of the range.
in	<i>last</i>	Iterator to the end of the range.
in	<i>pos</i>	Starting position of the copy.
in	<i>num</i>	Number of elements to copy.

Returns

Reference to the destination.

```
6.53.3.22 template<class VectorizedType , class , class GenericType , class MaskType , typename DummyType , class >
VectorizedType & magrathea::Vectorizer::set ( VectorizedType & destination, const GenericType & source, const
MaskType & mask, const DummyType = DummyType () ) [inline], [static]
```

Masked setter.

Copies elements of the source to the destination using a mask of boolean values : the values are copied only where the mask is true.

Template Parameters

<i>VectorizedType</i>	(Vectorized type.)
<i>GenericType</i>	(Generic type.)
<i>MaskType</i>	(Mask Type.)
<i>DummyType</i>	(Dummy parameter type.)

Parameters

<i>in, out</i>	<i>destination</i>	Destination of the copy.
<i>in</i>	<i>source</i>	Source of the copy.
<i>in</i>	<i>mask</i>	Boolean mask.

Returns

Reference to the destination.

```
6.53.3.23 template<class VectorizedType , class , typename DataType , class MaskType , typename DummyType , class >
    VectorizedType & magrathea::Vectorizer::set( VectorizedType & destination, const std::initializer_list< DataType >
& source, const MaskType & mask, const DummyType = DummyType () ) [inline], [static]
```

Masked list setter.

Copies the contents of the source to the destination using a mask of boolean values : the values are copied only where the mask is true. The iteration over values in the destination and in the source list are independant : the n-th element of the list is copied to the n-th true element of the destination. If the list is too small, empty values are added to its end.

Template Parameters

<i>VectorizedType</i>	(Vectorized type.)
<i>DataType</i>	(Data type.)
<i>MaskType</i>	(Mask Type.)
<i>DummyType</i>	(Dummy parameter type.)

Parameters

<i>in, out</i>	<i>destination</i>	Destination of the copy.
<i>in</i>	<i>source</i>	Source of the copy.
<i>in</i>	<i>mask</i>	Boolean mask.

Returns

Reference to the destination.

```
6.53.3.24 template<class VectorizedType , class , typename IteratorType , class MaskType , typename DummyType , class >
    VectorizedType & magrathea::Vectorizer::set( VectorizedType & destination, const IteratorType & first, const
IteratorType & last, const MaskType & mask, const DummyType = DummyType (), typename std::iterator_traits<
IteratorType >::iterator_category * = nullptr ) [inline], [static]
```

Masked range setter.

Copies the values from the range to the destination using a mask of boolean values : the values are copied only where the mask is true. The iteration over values in the destination and in the range list are independant : the n-th element of the range is copied to the n-th true element of the destination. The copy stops as soon as the end of the range is encountered or if the number of elements to copy is reached.

Template Parameters

<i>VectorizedType</i>	(Vectorized type.)
<i>Iterator</i>	(Iterator or pointer type.)
<i>MaskType</i>	(Mask Type.)
<i>DummyType</i>	(Dummy parameter type.)

Parameters

<i>in, out</i>	<i>destination</i>	Destination of the copy.
<i>in</i>	<i>first</i>	Iterator to the beginning of the range.
<i>in</i>	<i>last</i>	Iterator to the end of the range.
<i>in</i>	<i>mask</i>	Boolean mask.

Returns

Reference to the destination.

6.53.3.25 unsigned int magrathea::Vectorizer::size () const [inline]

Get the size of the container.

Returns the current number of elements. This function is required by the vectorization mechanism.

Returns

Copy of the size.

Exceptions

<i>std::logic_error</i>	This function should be overloaded by derived classes.
-------------------------	--

6.53.3.26 void magrathea::Vectorizer::type () const [inline]

Get the data type.

Returns a copy of the default value of the data type.

Returns

Copy of the default value of the data type.

Exceptions

<i>std::logic_error</i>	This function should be overloaded by derived classes.
-------------------------	--

The documentation for this class was generated from the following file:

- /data/home/mbreton/magrathea_pathfinder/src/magrathea/vectorizer.h

6.54 magrathea::Wrapper< Type > Exception Template Reference

Basic value wrapper with getter and setter.

```
#include <wrapper.h>
```

Public Member Functions**Lifecycle**

- template<typename OtherType = Type, class = typename std::enable_if<std::is_convertible<OtherType, Type>::value>::type> constexpr [Wrapper](#) (const OtherType &source=Type())

Implicit value constructor.

- template<typename OtherType = Type, class = typename std::enable_if<std::is_convertible<OtherType, Type>::value>::type> constexpr Wrapper (const Wrapper< OtherType > &source)

Implicit wrapper constructor.

Assignment

- template<typename OtherType = Type, class = typename std::enable_if<std::is_convertible<OtherType, Type>::value>::type> Wrapper< Type > & operator= (const OtherType &source)

Value assignment operator.

- template<typename OtherType = Type, class = typename std::enable_if<std::is_convertible<OtherType, Type>::value>::type> Wrapper< Type > & operator= (const Wrapper< OtherType > &source)

Wrapped assignment operator.

Operators

- constexpr operator Type () const

Cast operator.

- constexpr const Type & operator() () const

Immutable getter operator.

- Type & operator() ()

Getter operator.

- template<typename OtherType = Type, class = typename std::enable_if<std::is_convertible<OtherType, Type>::value>::type> Type & operator() (const OtherType &source)

Value setter operator.

- template<typename OtherType = Type, class = typename std::enable_if<std::is_convertible<OtherType, Type>::value>::type> Type & operator() (const Wrapper< OtherType > &source)

Wrapped setter operator.

Static Public Member Functions

Test

- static int example ()

Example function.

Public Attributes

Data members

- Type _data

Wrapped object.

6.54.1 Detailed Description

template<typename Type = double>exception magrathea::Wrapper< Type >

Basic value wrapper with getter and setter.

Provides a class that can wrap a value or an object and allows access by `operator()`. It can be used as a public class member to avoid the writing of trivial getter and setter.

Template Parameters

Type	Wrapped type.
------	---------------

6.54.2 Constructor & Destructor Documentation

6.54.2.1 `template<typename Type> template<typename OtherType, class> constexpr magrathea::Wrapper<Type>::Wrapper(const OtherType & source = Type())`

Implicit value constructor.

Implicitly constructs the wrapper from a value of a convertible type.

Template Parameters

<i>OtherType</i>	(Other data type.)
------------------	--------------------

Parameters

in	<i>source</i>	Value.
----	---------------	--------

6.54.2.2 `template<typename Type> template<typename OtherType, class> constexpr magrathea::Wrapper<Type>::Wrapper(const Wrapper<OtherType> & source)`

Implicit wrapper constructor.

Implicitly constructs the wrapper from a wrapper of a convertible type.

Template Parameters

<i>OtherType</i>	(Other data type.)
------------------	--------------------

Parameters

in	<i>source</i>	Other wrapper.
----	---------------	----------------

6.54.3 Member Function Documentation

6.54.3.1 `template<typename Type> int magrathea::Wrapper<Type>::example() [static]`

Example function.

Tests and demonstrates the use of [Wrapper](#).

Returns

0 if no error.

6.54.3.2 `template<typename Type> constexpr magrathea::Wrapper<Type>::operator Type() const`

Cast operator.

Implicitly casts the wrapper to the wrapped type.

Returns

Copy.

6.54.3.3 `template<typename Type> constexpr const Type & magrathea::Wrapper<Type>::operator()() const`

Immutable getter operator.

Returns a const reference to the wrapped object.

Returns

Const reference.

6.54.3.4 template<typename Type> Type & magrathea::Wrapper< Type >::operator()() [inline]

Getter operator.

Returns a reference to the wrapped object.

Returns

Reference.

6.54.3.5 template<typename Type> template<typename OtherType, class > Type & magrathea::Wrapper< Type >::operator()(const OtherType & source) [inline]

Value setter operator.

Sets the contents from a value of a convertible type.

Template Parameters

<i>OtherType</i>	(Other data type.)
------------------	--------------------

Parameters

in	<i>source</i>	Value.
----	---------------	--------

Returns

Reference.

6.54.3.6 template<typename Type> template<typename OtherType, class > Type & magrathea::Wrapper< Type >::operator()(const Wrapper< OtherType > & source) [inline]

Wrapped setter operator.

Sets the contents from a wrapper of a convertible type.

Template Parameters

<i>OtherType</i>	(Other data type.)
------------------	--------------------

Parameters

in	<i>source</i>	Other wrapper.
----	---------------	----------------

Returns

Reference.

6.54.3.7 `template<typename Type > template<typename OtherType , class > Wrapper< Type > & magrathea::Wrapper< Type >::operator= (const OtherType & source) [inline]`

Value assignment operator.

Assigns the contents from a value of a convertible type.

Template Parameters

<i>OtherType</i>	(Other data type.)
------------------	--------------------

Parameters

in	<i>source</i>	Value.
----	---------------	--------

Returns

Self reference.

6.54.3.8 `template<typename Type > template<typename OtherType , class > Wrapper< Type > & magrathea::Wrapper< Type >::operator= (const Wrapper< OtherType > & source) [inline]`

Wrapped assignment operator.

Assigns the contents from a wrapper of a convertible type.

Template Parameters

<i>OtherType</i>	(Other data type.)
------------------	--------------------

Parameters

in	<i>source</i>	Other wrapper.
----	---------------	----------------

Returns

Self reference.

6.54.4 Member Data Documentation

6.54.4.1 `template<typename Type = double> Type magrathea::Wrapper< Type >::_data`

Wrapped object.

The documentation for this exception was generated from the following file:

- /data/home/mbreton/mgrathea_pathfinder/src/mgrathea/wrapper.h

Chapter 7

File Documentation

7.1 /data/home/mbreton/magrathea_pathfinder/src/catalogues.cpp File Reference

Main function of the raytracer to create catalogues.

```
#include <ctime>
#include <iostream>
#include <iomanip>
#include <fstream>
#include <algorithm>
#include <utility>
#include <random>
#include <string>
#include <vector>
#include <deque>
#include <array>
#include <tuple>
#include <atomic>
#include <mutex>
#include <mpi.h>
#include "magrathea/simplehyperoctree.h"
#include "magrathea/simplehyperoctreeindex.h"
#include "magrathea/hypersphere.h"
#include "magrathea/hypercube.h"
#include "magrathea/constants.h"
#include "magrathea/evolution.h"
#include "magrathea/timer.h"
#include "cone.h"
#include "utility.h"
#include "input.h"
#include "output.h"
#include "integrator.h"
#include "miscellaneous.h"
#include "catalogues.h"
#include "gravity.h"
#include "TReadHDF5.h"
```

Functions

- int **main** (int argc, char *argv[])

Main function.

7.1.1 Detailed Description

Main function of the raytracer to create catalogues.

Author

Vincent Reverdy (vince.rev@gmail.com), Michel-Andrès Breton (michel-andres.breton@obspm.fr)

Date

2012-2013, 2015-2021

Copyright

CECILL-B License

7.1.2 Function Documentation

7.1.2.1 int main (int *argc*, char * *argv*[])

Main function.

Main function of the program. It takes a namelist as a parameter and run the raytracing using MPI parallelization. To compile it, use make To run it, use: mpirun -np NTASKS ./raytracer raytracer.txt

Parameters

in	<i>argc</i>	Number of arguments.
in	<i>argv</i>	List of arguments.

Returns

Zero on success, error code otherwise.

7.2 /data/home/mbreton/magrathea_pathfinder/src/catalogues.h File Reference

Some catalogues function.

```
#include <ctime>
#include <iostream>
#include <iomanip>
#include <fstream>
#include <algorithm>
#include <utility>
#include <random>
#include <string>
#include <vector>
#include <deque>
#include <array>
#include <tuple>
#include <atomic>
#include <mutex>
#include <mpi.h>
#include "magrathea/simplehyperoctree.h"
#include "magrathea/simplehyperoctreeindex.h"
#include "magrathea/hypersphere.h"
#include "magrathea/abstracthypersphere.h"
#include "magrathea/hypercube.h"
#include "magrathea/constants.h"
#include "magrathea/evolution.h"
#include "magrathea/timer.h"
#include "cone.h"
#include "utility.h"
#include "input.h"
#include "output.h"
#include "TReadHDF5.h"
#include "lensing.h"
```

Classes

- struct [parameters_t](#)
- class [Catalogues](#)

Variables

- struct [parameters_t](#) [parameters](#)

7.2.1 Detailed Description

Some catalogues function.

Author

Vincent Reverdy (vince.rev@gmail.com), Michel-Andrès Breton (michel-andres.breton@obspm.fr)

Date

2012-2013, 2015-2021

Copyright

CECILL-B License

7.2.2 Variable Documentation

7.2.2.1 struct parameters_t parameters

7.3 /data/home/mbreton/magrathea_pathfinder/src/cone.h File Reference

Three-dimensional cone.

```
#include <iostream>
#include <iomanip>
#include <cmath>
#include <type_traits>
#include <array>
#include <utility>
#include "magrathea/abstractshape.h"
#include "magrathea/abstractsubstance.h"
```

Classes

- exception [Cone< Vector, Scalar >](#)

Three-dimensional cone.

7.3.1 Detailed Description

Three-dimensional cone.

Author

Vincent Reverdy (vince.rev@gmail.com), Michel-Andrès Breton (michel-andres.breton@obspm.fr)

Date

2012-2013, 2015-2021

Copyright

CECILL-B License

7.4 /data/home/mbreton/magrathea_pathfinder/src/create_octree.cpp File Reference

```
#include <ctime>
#include <iostream>
#include <iomanip>
#include <fstream>
#include <algorithm>
#include <utility>
#include <random>
#include <string>
#include <vector>
#include <deque>
#include <array>
#include <tuple>
#include <atomic>
#include <mutex>
#include <mpi.h>
#include "magrathea/simplehyperoctree.h"
#include "magrathea/simplehyperoctreeindex.h"
#include "magrathea/hypersphere.h"
#include "magrathea/hypercube.h"
#include "magrathea/constants.h"
#include "magrathea/evolution.h"
#include "magrathea/timer.h"
#include "cone.h"
#include "utility.h"
#include "input.h"
#include "output.h"
#include "integrator.h"
#include "miscellaneous.h"
#include "create_octree.h"
#include "gravity.h"
#include "TReadHDF5.h"
```

Functions

- int **main** (int argc, char *argv[])

Main function.

7.4.1 Function Documentation**7.4.1.1 int main (int argc, char * argv[])**

Main function.

Main function of the program. It takes a namelist as a parameter and run the raytracing using MPI parallelization. To compile it, use make To run it, use: mpirun -np NTASKS ./raytracer raytracer.txt

Parameters

in	<i>argc</i>	Number of arguments.
in	<i>argv</i>	List of arguments.

Returns

Zero on success, error code otherwise.

7.5 /data/home/mbreton/magrathea_pathfinder/src/create_octree.h File Reference

Some create_octree function.

```
#include <ctime>
#include <iostream>
#include <iomanip>
#include <fstream>
#include <algorithm>
#include <utility>
#include <random>
#include <string>
#include <vector>
#include <deque>
#include <array>
#include <tuple>
#include <atomic>
#include <mutex>
#include <mpi.h>
#include "magrathea/simplehyperoctree.h"
#include "magrathea/simplehyperoctreeindex.h"
#include "magrathea/hypersphere.h"
#include "magrathea/abstracthypersphere.h"
#include "magrathea/hypercube.h"
#include "magrathea/constants.h"
#include "magrathea/evolution.h"
#include "magrathea/timer.h"
#include "cone.h"
#include "utility.h"
#include "input.h"
#include "output.h"
#include "TReadHDF5.h"
```

Classes

- struct [parameters_t](#)
- class [Create_octree](#)

Variables

- struct [parameters_t](#) [parameters](#)

7.5.1 Detailed Description

Some create_octree function.

Author

Vincent Reverdy (vince.rev@gmail.com), Michel-Andr s Breton (michel-andres.breton@obspm.fr)

Date

2012-2013, 2015-2021

Copyright

CECILL-B License

7.5.2 Variable Documentation

7.5.2.1 struct parameters_t parameters

7.6 /data/home/mbreton/magrathea_pathfinder/src/gravity.h File Reference

[Gravity](#) cell implementation for raytracing.

```
#include <iostream>
#include <iomanip>
#include <type_traits>
#include <array>
#include <utility>
#include "magrathea/abstractcontents.h"
#include "magrathea/euleriancategory.h"
```

Classes

- exception [Gravity< Type, Dimension >](#)

Gravity cell implementation for raytracing.

7.6.1 Detailed Description

[Gravity](#) cell implementation for raytracing.

Author

Vincent Reverdy (vince.rev@gmail.com), Michel-Andr s Breton (michel-andres.breton@obspm.fr)

Date

2012-2013, 2015-2021

Copyright

CECILL-B License

7.7 /data/home/mbreton/magrathea_pathfinder/src/gravity2.h File Reference

[Gravity](#) cell implementation for raytracing.

```
#include <iostream>
#include <iomanip>
#include <type_traits>
#include <array>
#include <utility>
#include "magrathea/abstractcontents.h"
#include "magrathea/euleriancategory.h"
```

Classes

- exception [Gravity< Type, Dimension >](#)

Gravity cell implementation for raytracing.

7.7.1 Detailed Description

[Gravity](#) cell implementation for raytracing.

Author

Vincent Reverdy (vince.rev@gmail.com), Michel-Andr s Breton (michel-andres.breton@obspm.fr)

Date

2012-2013, 2015-2021

Copyright

CECILL-B License

7.8 /data/home/mbreton/magrathea_pathfinder/src/hmaps.cpp File Reference

Main function of the raytracer to create Healpix maps.

```
#include <ctime>
#include <iostream>
#include <iomanip>
#include <fstream>
#include <algorithm>
#include <utility>
#include <random>
#include <string>
#include <vector>
#include <deque>
#include <array>
#include <tuple>
#include <atomic>
#include <mutex>
#include <mpi.h>
#include "magrathea/simplehyperoctree.h"
#include "magrathea/simplehyperoctreeindex.h"
#include "magrathea/hypersphere.h"
#include "magrathea/hypercube.h"
#include "magrathea/constants.h"
#include "magrathea/evolution.h"
#include "magrathea/timer.h"
#include "cone.h"
#include "utility.h"
#include "input.h"
#include "output.h"
#include "integrator.h"
#include "miscellaneous.h"
#include "hmaps.h"
#include "gravity.h"
#include "TReadHDF5.h"
#include <chealpix.h>
#include <pointing.h>
#include <healpix_map.h>
```

Functions

- int **main** (int argc, char *argv[])

Main function.

7.8.1 Detailed Description

Main function of the raytracer to create Healpix maps.

Author

Vincent Reverdy (vince.rev@gmail.com), Michel-Andr s Breton (michel-andres.breton@obspm.fr)

Date

2012-2013, 2015-2021

Copyright

CECILL-B License

7.8.2 Function Documentation

7.8.2.1 int main (int *argc*, char * *argv*[])

Main function.

Main function of the program. It takes a namelist as a parameter and run the raytracing using MPI parallelization. To compile it, use make To run it, use: mpirun -np NTASKS ./raytracer raytracer.txt

Parameters

in	<i>argc</i>	Number of arguments.
in	<i>argv</i>	List of arguments.

Returns

Zero on success, error code otherwise.

7.9 /data/home/mbreton/magrathea_pathfinder/src/hmaps.h File Reference

Some hmaps function.

```
#include <ctime>
#include <iostream>
#include <iomanip>
#include <fstream>
#include <algorithm>
#include <utility>
#include <random>
#include <string>
#include <vector>
#include <deque>
#include <array>
#include <tuple>
#include <atomic>
#include <mutex>
#include <mpi.h>
#include "magrathea/simplehyperoctree.h"
#include "magrathea/simplehyperoctreeindex.h"
#include "magrathea/hypersphere.h"
#include "magrathea/abstracthypersphere.h"
#include "magrathea/hypercube.h"
#include "magrathea/constants.h"
#include "magrathea/evolution.h"
#include "magrathea/timer.h"
#include "cone.h"
#include "utility.h"
#include "input.h"
#include "output.h"
#include "TReadHDF5.h"
#include "lensing.h"
#include <chealpix.h>
#include <pointing.h>
#include <healpix_map.h>
```

Classes

- struct [parameters_t](#)
- class [Hmaps](#)

Variables

- struct [parameters_t](#) [parameters](#)

7.9.1 Detailed Description

Some hmaps function.

Author

Vincent Reverdy (vince.rev@gmail.com), Michel-Andr s Breton (michel-andres.breton@obspm.fr)

Date

2012-2013, 2015-2021

Copyright

CECILL-B License

7.9.2 Variable Documentation

7.9.2.1 struct parameters_t parameters

7.10 /data/home/mbreton/magrathea_pathfinder/src/input.h File Reference

[Input](#) utilities for raytracing.

```
#include <iostream>
#include <iomanip>
#include <fstream>
#include <sstream>
#include <type_traits>
#include <algorithm>
#include <thread>
#include <limits>
#include <string>
#include <vector>
#include <array>
#include <tuple>
#include <map>
#include <cmath>
#include <cctype>
#include <mutex>
#include <atomic>
#include "magrathea/simplehyperoctree.h"
#include "magrathea/simplehyperoctreeindex.h"
#include "magrathea/filesystem.h"
#include "magrathea/filelist.h"
#include "magrathea/datahandler.h"
#include "magrathea/constants.h"
#include "magrathea/hypercube.h"
#include "magrathea/hypersphere.h"
#include "cone.h"
#include "utility.h"
#include "gravity.h"
#include "photon.h"
#include "TReadHDF5.h"
#include <unordered_set>
```

Classes

- exception [Input](#)

[Input](#) utilities for raytracing.

7.10.1 Detailed Description

[Input](#) utilities for raytracing.

Author

Vincent Reverdy (vince.rev@gmail.com), Michel-Andrès Breton (michel-andres.breton@obspm.fr)

Date

2012-2013, 2015-2021

Copyright

CECILL-B License

7.11 /data/home/mbreton/magrathea_pathfinder/src/integrator.h File Reference

Integration utilities for raytracing.

```
#include <iostream>
#include <iomanip>
#include <fstream>
#include <type_traits>
#include <limits>
#include <string>
#include <vector>
#include <tuple>
#include <array>
#include <cmath>
#include <random>
#include "magrathea/simplehyperoctree.h"
#include "magrathea/simplehyperoctreeindex.h"
#include "magrathea/constants.h"
#include "magrathea/hypersphere.h"
#include "magrathea/evolution.h"
#include "utility.h"
#include "gravity.h"
#include "photon.h"
#include "cone.h"
#include "output.h"
```

Classes

- exception [Integrator](#)

Integration utilities for raytracing.

7.11.1 Detailed Description

Integration utilities for raytracing.

Author

Vincent Reverdy (vince.rev@gmail.com), Michel-Andrès Breton (michel-andres.breton@obspm.fr)

Date

2012-2013, 2015-2021

Copyright

CECILL-B License

7.12 /data/home/mbreton/magrathea_pathfinder/src/lensing.h File Reference

Integration utilities for raytracing.

```
#include <iostream>
#include <iomanip>
#include <fstream>
#include <type_traits>
#include <limits>
#include <string>
#include <vector>
#include <tuple>
#include <array>
#include <cmath>
#include <random>
#include "magrathea/simplehyperoctree.h"
#include "magrathea/simplehyperoctreeindex.h"
#include "magrathea/constants.h"
#include "magrathea/hypersphere.h"
#include "magrathea/evolution.h"
#include "utility.h"
#include "gravity.h"
#include "photon.h"
#include "cone.h"
#include "output.h"
```

Classes

- class [Lensing](#)

Integration utilities for raytracing.

7.12.1 Detailed Description

Integration utilities for raytracing.

Author

Vincent Reverdy (vince.rev@gmail.com), Michel-Andrès Breton (michel-andres.breton@obspm.fr)

Date

2012-2013, 2015-2021

Copyright

CECILL-B License

7.13 /data/home/mbreton/magrathea_pathfinder/src/magrathea/aboutinstitute.h File Reference

Information about an institution or an organization.

```
#include <iostream>
#include <iomanip>
#include <utility>
#include <string>
#include "abstractaboutobject.h"
```

Classes

- exception [magrathea::AboutInstitute](#)

Information about an institution or an organization.

Namespaces

- namespace [magrathea](#)

7.13.1 Detailed Description

Information about an institution or an organization.

Author

Vincent Reverdy (vince.rev@gmail.com)

Date

2012-2013

Copyright

CECILL-B License

7.14 /data/home/mbreton/magrathea_pathfinder/src/magrathea/aboutlicense.h File Reference

Information about the license of a code.

```
#include <iostream>
#include <iomanip>
#include <utility>
#include <string>
#include <array>
#include <initializer_list>
#include "abstractaboutobject.h"
```

Classes

- exception [magrathea::AboutLicense](#)
Information about the license of a code.

Namespaces

- namespace [magrathea](#)

7.14.1 Detailed Description

Information about the license of a code.

Author

Vincent Reverdy (vince.rev@gmail.com)

Date

2012-2013

Copyright

CECILL-B License

7.15 /data/home/mbreton/magrathea_pathfinder/src/magrathea/aboutobject.h File Reference

Basic about object with information on something.

```
#include <iostream>
#include <iomanip>
#include <type_traits>
#include <typeinfo>
#include <utility>
#include <tuple>
#include <array>
#include <ratio>
#include <string>
#include <sstream>
#include "abstractaboutobject.h"
```

Classes

- exception [magrathea::AboutObject< Types >](#)
Basic about object with information on something.

Namespaces

- namespace [magrathea](#)

7.15.1 Detailed Description

Basic about object with information on something.

Author

Vincent Reverdy (vince.rev@gmail.com)

Date

2012-2013

Copyright

CECILL-B License

7.16 /data/home/mbreton/magrathea_pathfinder/src/magrathea/aboutpeople.h File Reference

Information about a developer, an author, or a contributor.

```
#include <iostream>
#include <iomanip>
#include <utility>
#include <string>
#include "abstractaboutobject.h"
```

Classes

- exception [magrathea::AboutPeople](#)

Information about a developer, an author, or a contributor.

Namespaces

- namespace [magrathea](#)

7.16.1 Detailed Description

Information about a developer, an author, or a contributor.

Author

Vincent Reverdy (vince.rev@gmail.com)

Date

2012-2013

Copyright

CECILL-B License

7.17 /data/home/mbreton/magrathea_pathfinder/src/magrathea/abstractaboutobject.h File Reference

Tuple abstraction of generic about object.

```
#include <iostream>
#include <iomanip>
#include <type_traits>
#include <typeinfo>
#include <utility>
#include <tuple>
#include <array>
#include <ratio>
#include <string>
#include <sstream>
```

Classes

- class [magrathea::AbstractAboutObject< Crtp, Types >](#)

Tuple abstraction of generic about object.

Namespaces

- namespace [magrathea](#)

Functions

- template<class SelfCrtp , class... SelfTypes>
`std::ostream & magrathea::operator<< (std::ostream &lhs, const AbstractAboutObject< SelfCrtp, SelfTypes...> &rhs)`

Output stream operator.

7.17.1 Detailed Description

Tuple abstraction of generic about object.

Author

Vincent Reverdy (vince.rev@gmail.com)

Date

2012-2013

Copyright

CECILL-B License

7.18 /data/home/mbreton/magrathea_pathfinder/src/magrathea/abstractcontents.h File Reference

Tuple abstraction of numerical simulation contents.

```
#include <iostream>
#include <iomanip>
#include <type_traits>
#include <typeinfo>
#include <utility>
#include <tuple>
#include <array>
#include <ratio>
#include <string>
#include <sstream>
```

Classes

- class [magrathea::AbstractContents< Crtp, Category, Types >](#)

Tuple abstraction of numerical simulation contents.

Namespaces

- namespace [magrathea](#)

Functions

- template<class SelfCrtp , class SelfCategory , class... SelfTypes>
std::ostream & [magrathea::operator<<](#) (std::ostream &lhs, const AbstractContents< SelfCrtp, SelfCategory, SelfTypes...> &rhs)

Output stream operator.

7.18.1 Detailed Description

Tuple abstraction of numerical simulation contents.

Author

Vincent Reverdy (vince.rev@gmail.com)

Date

2012-2013

Copyright

CECILL-B License

7.19 /data/home/mbreton/magrathea_pathfinder/src/magrathea/abstracthypercube.h File Reference

Abstract function provider for n-dimensional cubes.

```
#include <iostream>
#include <iomanip>
#include <type_traits>
#include <random>
#include <algorithm>
#include "abstractshape.h"
```

Classes

- class [magrathea::AbstractHyperCube< Crtp, Dimension, Vector, Scalar >](#)
Abstract function provider for n-dimensional cubes.

Namespaces

- namespace [magrathea](#)

7.19.1 Detailed Description

Abstract function provider for n-dimensional cubes.

Author

Vincent Reverdy (vince.rev@gmail.com)

Date

2012-2013

Copyright

CECILL-B License

7.20 /data/home/mbreton/magrathea_pathfinder/src/magrathea/abstracthypersphere.h File Reference

Abstract function provider for n-dimensional spheres.

```
#include <iostream>
#include <iomanip>
#include <type_traits>
#include <random>
#include <algorithm>
#include "abstractshape.h"
```

Classes

- class [magrathea::AbstractHyperSphere< Crtp, Dimension, Vector, Scalar >](#)
Abstract function provider for n-dimensional spheres.

Namespaces

- namespace [magrathea](#)

7.20.1 Detailed Description

Abstract function provider for n-dimensional spheres.

Author

Vincent Reverdy (vince.rev@gmail.com)

Date

2012-2013

Copyright

CECILL-B License

7.21 /data/home/mbreton/magrathea_pathfinder/src/magrathea/abstractnarray.h File Reference

Abstract base class of n-dimensional mathematical arrays.

```
#include <iostream>
#include <iomanip>
#include <type_traits>
#include <stdexcept>
#include <initializer_list>
#include <array>
#include <iterator>
#include <random>
#include <cstdlib>
#include <cmath>
#include <algorithm>
#include <functional>
#include <utility>
#include "staticvectorizer.h"
```

Classes

- class [magrathea::AbstractNArray< Kind, Size, Crtp, Type, Parameters >](#)

Abstract base class of n-dimensional mathematical arrays.

Namespaces

- namespace [magrathea](#)

7.21.1 Detailed Description

Abstract base class of n-dimensional mathematical arrays.

Author

Vincent Reverdy (vince.rev@gmail.com)

Date

2012-2013

Copyright

CECILL-B License

7.22 /data/home/mbreton/magrathea_pathfinder/src/magrathea/abstractshape.h File Reference

Common abstraction of n-dimensional shapes.

```
#include <iostream>
#include <iomanip>
#include <type_traits>
#include <limits>
#include <cmath>
```

Classes

- class [magrathea::AbstractShape](#)
Common abstraction of n-dimensional shapes.

Namespaces

- namespace [magrathea](#)

7.22.1 Detailed Description

Common abstraction of n-dimensional shapes.

Author

Vincent Reverdy (vince.rev@gmail.com)

Date

2012-2013

Copyright

CECILL-B License

7.23 /data/home/mbreton/magrathea_pathfinder/src/magrathea/abstractstep.h File Reference

Abstraction of an evolution step.

```
#include <iostream>
#include <iomanip>
#include <type_traits>
#include <typeinfo>
#include <utility>
#include <tuple>
#include <array>
#include <ratio>
#include <string>
#include <sstream>
```

Classes

- class [magrathea::AbstractStep< Crtp, Scalar, Array, Tuple >](#)

Abstraction of an evolution step.

Namespaces

- namespace [magrathea](#)

Functions

- template<class SelfCrtp , class SelfScalar , class SelfArray , class SelfTuple >
std::ostream & [magrathea::operator<<](#) (std::ostream &lhs, const AbstractStep< SelfCrtp, SelfScalar, SelfArray, SelfTuple > &rhs)

Output stream operator.

7.23.1 Detailed Description

Abstraction of an evolution step.

Author

Vincent Reverdy (vince.rev@gmail.com)

Date

2012-2013

Copyright

CECILL-B License

7.24 /data/home/mbreton/magrathea_pathfinder/src/magrathea/abstractsubstance.h File Reference

Tuple abstraction of geometrical substance.

```
#include <iostream>
#include <iomanip>
#include <type_traits>
#include <typeinfo>
#include <utility>
#include <tuple>
#include <array>
#include <ratio>
#include <string>
#include <sstream>
```

Classes

- class [magrathea::AbstractSubstance< Crtp, Types >](#)
Tuple abstraction of geometrical substance.

Namespaces

- namespace [magrathea](#)

Functions

- template<class SelfCrtp , class... SelfTypes>
`std::ostream & magrathea::operator<< (std::ostream &lhs, const AbstractSubstance< SelfCrtp, SelfTypes...> &rhs)`
Output stream operator.

7.24.1 Detailed Description

Tuple abstraction of geometrical substance.

Author

Vincent Reverdy (vince.rev@gmail.com)

Date

2012-2013

Copyright

CECILL-B License

7.25 /data/home/mbreton/magrathea_pathfinder/src/magrathea/arrayfile.h File Reference

Handler for text files representing arrays of data.

```
#include <iostream>
#include <iomanip>
```

Namespaces

- namespace [magrathea](#)

7.25.1 Detailed Description

Handler for text files representing arrays of data.

Author

Vincent Reverdy (vince.rev@gmail.com)

Date

2012-2013

Copyright

CECILL-B License

7.26 /data/home/mbreton/magrathea_pathfinder/src/magrathea/basefile.h File Reference

Base class for input and output with files.

```
#include <iostream>
#include <iomanip>
```

Namespaces

- namespace [magrathea](#)

7.26.1 Detailed Description

Base class for input and output with files.

Author

Vincent Reverdy (vince.rev@gmail.com)

Date

2012-2013

Copyright

CECILL-B License

7.27 /data/home/mbreton/magrathea_pathfinder/src/magrathea/binaryfile.h File Reference

Base class to manage generic binary files.

```
#include <iostream>
#include <iomanip>
```

Namespaces

- namespace [magrathea](#)

7.27.1 Detailed Description

Base class to manage generic binary files.

Author

Vincent Reverdy (vince.rev@gmail.com)

Date

2012-2013

Copyright

CECILL-B License

7.28 /data/home/mbreton/magrathea_pathfinder/src/magrathea/constant.h File Reference

Numerical constant with `constexpr` constructor.

```
#include <iostream>
#include <iomanip>
#include <type_traits>
#include <array>
#include <ratio>
#include <cmath>
```

Classes

- exception [magrathea::Constant< Type, Size >](#)
Numerical constant with `constexpr` constructor.

Namespaces

- namespace [magrathea](#)

Functions

- template<typename SelfType , unsigned int SelfSize>
`std::ostream & magrathea::operator<< (std::ostream &lhs, const Constant< SelfType, SelfSize > &rhs)`
Output stream operator.

7.28.1 Detailed Description

Numerical constant with `constexpr` constructor.

Author

Vincent Reverdy (vince.rev@gmail.com)

Date

2012-2013

Copyright

CECILL-B License

7.29 /data/home/mbreton/magrathea_pathfinder/src/magrathea/constants.h File Reference

Common mathematical and physical constants.

```
#include <iostream>
#include <iomanip>
#include "constant.h"
```

Classes

- exception [magrathea::Constants< Type >](#)
Common mathematical and physical constants.

Namespaces

- namespace [magrathea](#)

7.29.1 Detailed Description

Common mathematical and physical constants.

Author

Vincent Reverdy (vince.rev@gmail.com)

Date

2012-2013

Copyright

CECILL-B License

7.30 /data/home/mbreton/magrathea_pathfinder/src/magrathea/contents.h File Reference

Basic implementation of numerical simulation contents.

```
#include <iostream>
#include <iomanip>
#include <type_traits>
#include <typeinfo>
#include <utility>
#include <tuple>
#include <array>
#include <ratio>
#include <string>
#include <sstream>
#include "abstractcontents.h"
```

Classes

- exception [magrathea::Contents< Category, Types >](#)

Basic implementation of numerical simulation contents.

Namespaces

- namespace [magrathea](#)

7.30.1 Detailed Description

Basic implementation of numerical simulation contents.

Author

Vincent Reverdy (vince.rev@gmail.com)

Date

2012-2013

Copyright

CECILL-B License

7.31 /data/home/mbreton/magrathea_pathfinder/src/magrathea/datahandler.h File Reference

Set of basic operations on binary data related to IO.

```
#include <iostream>
#include <iomanip>
#include <fstream>
#include <sstream>
#include <string>
#include <cstdlib>
#include <cstdio>
#include <cstring>
#include <type_traits>
#include <algorithm>
#include <limits>
#include <iterator>
#include <array>
#include <tuple>
#include <vector>
#include <set>
```

Classes

- exception [magrathea::DataHandler](#)
Set of basic operations on binary data related to IO.

Namespaces

- namespace [magrathea](#)

7.31.1 Detailed Description

Set of basic operations on binary data related to IO.

Author

Vincent Reverdy (vince.rev@gmail.com)

Date

2012-2013

Copyright

CECILL-B License

7.32 /data/home/mbreton/magrathea_pathfinder/src/magrathea/datamodel.h File Reference

Management of fundamental types representation.

```
#include <iostream>
#include <iomanip>
#include <type_traits>
#include <stdexcept>
#include <limits>
```

Classes

- exception [magrathea::DataModel](#)

Management of fundamental types representation.

Namespaces

- namespace [magrathea](#)

Functions

- std::ostream & [magrathea::operator<<](#) (std::ostream &lhs, const DataModel &rhs)

Output stream operator.

7.32.1 Detailed Description

Management of fundamental types representation.

Author

Vincent Reverdy (vince.rev@gmail.com)

Date

2012-2013

Copyright

CECILL-B License

7.33 /data/home/mbreton/magrathea_pathfinder/src/magrathea/datasize.h File Reference

Wrapper of binary data size and manager of unit conversion.

```
#include <iostream>
#include <iomanip>
#include <type_traits>
#include <stdexcept>
#include <ratio>
#include <limits>
#include <string>
```

Classes

- exception [magrathea::DataSize](#)

Wrapper of binary data size and manager of unit conversion.

Namespaces

- namespace [magrathea](#)

Functions

- std::ostream & [magrathea::operator<<](#) (std::ostream &lhs, const DataSize &rhs)
Output stream operator.

7.33.1 Detailed Description

Wrapper of binary data size and manager of unit conversion.

Author

Vincent Reverdy (vince.rev@gmail.com)

Date

2012-2013

Copyright

CECILL-B License

7.34 /data/home/mbreton/magrathea_pathfinder/src/magrathea/euleriancategory.h File Reference

Category concept of eulerian : data at a fixed position.

```
#include <iostream>
#include <iomanip>
```

Classes

- exception [magrathea::EulerianCategory](#)
Category concept of eulerian : data at a fixed position.

Namespaces

- namespace [magrathea](#)

7.34.1 Detailed Description

Category concept of eulerian : data at a fixed position.

Author

Vincent Reverdy (vince.rev@gmail.com)

Date

2012-2013

Copyright

CECILL-B License

7.35 /data/home/mbreton/magrathea_pathfinder/src/magrathea/evolution.h File Reference

Resizable container of steps dedicated to integration.

```
#include <iostream>
#include <iomanip>
#include <utility>
#include <type_traits>
#include <initializer_list>
#include <stdexcept>
#include <functional>
#include <algorithm>
#include <vector>
#include <string>
#include "step.h"
```

Classes

- exception [magrathea::Evolution< Type, Container >](#)

Resizable container of steps dedicated to integration.

Namespaces

- namespace [magrathea](#)

Functions

- template<class SelfType , class SelfContainer >
std::ostream & [magrathea::operator<<](#) (std::ostream &lhs, const Evolution< SelfType, SelfContainer > &rhs)

Output stream operator.

7.35.1 Detailed Description

Resizable container of steps dedicated to integration.

Date

2012-2013

Copyright

CECILL-B License

7.36 /data/home/mbreton/magrathea_pathfinder/src/magrathea/filelist.h File Reference

List of files based on a function or a vector.

```
#include <iostream>
#include <iomanip>
#include <cstdio>
#include <cctype>
#include <type_traits>
#include <stdexcept>
#include <functional>
#include <algorithm>
#include <initializer_list>
#include <array>
#include <vector>
#include <string>
#include <limits>
```

Classes

- exception [magrathea::FileList](#)

List of files based on a function or a vector.

Namespaces

- namespace [magrathea](#)

Functions

- std::ostream & [magrathea::operator<<](#) (std::ostream &lhs, const FileList &rhs)

Output stream operator.

7.36.1 Detailed Description

List of files based on a function or a vector.

Author

Vincent Reverdy (vince.rev@gmail.com)

Date

2012-2013

Copyright

CECILL-B License

7.37 /data/home/mbreton/magrathea_pathfinder/src/magrathea/filesystem.h File Reference

Global file management.

```
#include <iostream>
#include <iomanip>
#include <type_traits>
#include <stdexcept>
#include <fstream>
#include <sstream>
#include <cstdio>
#include <cstdint>
#include <ctime>
#include <chrono>
#include <string>
#include <vector>
#include <random>
#include <algorithm>
#include <limits>
```

Classes

- exception [magrathea::FileSystem](#)

Global file management.

Namespaces

- namespace [magrathea](#)

7.37.1 Detailed Description

Global file management.

Author

Vincent Reverdy (vince.rev@gmail.com)

Date

2012-2013

Copyright

CECILL-B License

7.38 /data/home/mbreton/magrathea_pathfinder/src/magrathea/fortranfile.h File Reference

Handler for binary files in fortran format.

```
#include <iostream>
#include <iomanip>
```

Namespaces

- namespace [magrathea](#)

7.38.1 Detailed Description

Handler for binary files in fortran format.

Author

Vincent Reverdy (vince.rev@gmail.com)

Date

2012-2013

Copyright

CECILL-B License

7.39 /data/home/mbreton/magrathea_pathfinder/src/magrathea/gridcategory.h File Reference

Category concept of grid : data related to the mesh.

```
#include <iostream>
#include <iomanip>
```

Classes

- exception [magrathea::GridCategory](#)
Category concept of grid : data related to the mesh.

Namespaces

- namespace [magrathea](#)

7.39.1 Detailed Description

Category concept of grid : data related to the mesh.

Author

Vincent Reverdy (vince.rev@gmail.com)

Date

2012-2013

Copyright

CECILL-B License

7.40 /data/home/mbreton/magrathea_pathfinder/src/magrathea/hypercube.h File Reference

N-dimensional cube.

```
#include <iostream>
#include <iomanip>
#include <type_traits>
#include <array>
#include <utility>
#include "abstractsubstance.h"
#include "abstracthypercube.h"
```

Classes

- exception [magrathea::HyperCube< Dimension, Vector, Scalar >](#)
N-dimensional cube.

Namespaces

- namespace [magrathea](#)

7.40.1 Detailed Description

N-dimensional cube.

Author

Vincent Reverdy (vince.rev@gmail.com)

Date

2012-2013

Copyright

CECILL-B License

7.41 /data/home/mbreton/magrathea_pathfinder/src/magrathea/hypersphere.h File Reference

N-dimensional sphere.

```
#include <iostream>
#include <iomanip>
#include <type_traits>
#include <array>
#include <utility>
#include "abstractsubstance.h"
#include "abstracthypersphere.h"
```

Classes

- exception [magrathea::HyperSphere< Dimension, Vector, Scalar >](#)
N-dimensional sphere.

Namespaces

- namespace [magrathea](#)

7.41.1 Detailed Description

N-dimensional sphere.

Author

Vincent Reverdy (vince.rev@gmail.com)

Date

2012-2013

Copyright

CECILL-B License

7.42 /data/home/mbreton/magrathea_pathfinder/src/magrathea/lagrangiancategory.h File Reference

Category concept of lagrangian : data moving with the flow.

```
#include <iostream>
#include <iomanip>
```

Classes

- exception [magrathea::LagrangianCategory](#)
Category concept of lagrangian : data moving with the flow.

Namespaces

- namespace [magrathea](#)

7.42.1 Detailed Description

Category concept of lagrangian : data moving with the flow.

Author

Vincent Reverdy (vince.rev@gmail.com)

Date

2012-2013

Copyright

CECILL-B License

7.43 /data/home/mbreton/magrathea_pathfinder/src/magrathea/logfile.h File Reference

Handler for text files representing logs.

```
#include <iostream>
#include <iomanip>
```

Namespaces

- namespace [magrathea](#)

7.43.1 Detailed Description

Handler for text files representing logs.

Author

Vincent Revardy (vince.rev@gmail.com)

Date

2012-2013

Copyright

CECILL-B License

7.44 /data/home/mbreton/magrathea_pathfinder/src/magrathea/namelistfile.h File Reference

Handler for text files representing lists of parameters.

```
#include <iostream>
#include <iomanip>
```

Namespaces

- namespace [magrathea](#)

7.44.1 Detailed Description

Handler for text files representing lists of parameters.

Author

Vincent Reverdy (vince.rev@gmail.com)

Date

2012-2013

Copyright

CECILL-B License

7.45 /data/home/mbreton/magrathea_pathfinder/src/magrathea/narray.h File Reference

Basic n-dimensional mathematical array.

```
#include <iostream>
#include <iomanip>
#include <type_traits>
#include <initializer_list>
#include <algorithm>
#include <random>
#include <cmath>
#include "abstractnarray.h"
```

Classes

- exception [magrathea::NArray< Type, Size >](#)
Basic n-dimensional mathematical array.

Namespaces

- namespace [magrathea](#)

7.45.1 Detailed Description

Basic n-dimensional mathematical array.

Author

Vincent Reverdy (vince.rev@gmail.com)

Date

2012-2013

Copyright

CECILL-B License

7.46 /data/home/mbreton/magrathea_pathfinder/src/magrathea/shape.h File Reference

Basic implementation of shape.

```
#include <iostream>
#include <iomanip>
#include "abstractshape.h"
```

Classes

- exception [magrathea::Shape](#)

Basic implementation of shape.

Namespaces

- namespace [magrathea](#)

7.46.1 Detailed Description

Basic implementation of shape.

Author

Vincent Reverdy (vince.rev@gmail.com)

Date

2012-2013

Copyright

CECILL-B License

7.47 /data/home/mbreton/magrathea_pathfinder/src/magrathea/simplehyperoctree.h File Reference

A simple hyperoctree based on bit manipulations.

```
#include <iostream>
#include <iomanip>
#include <sstream>
#include <type_traits>
#include <cstdlib>
#include <cmath>
#include <limits>
#include <algorithm>
#include <vector>
#include <set>
#include <array>
#include <tuple>
#include "simplehyperoctreeindex.h"
#include "../utility.h"
```

Classes

- exception `magrathea::SimpleHyperOctree< Type, Index, Data, Dimension, Position, Extent, Element, Container >`

A simple hyperoctree based on bit manipulations.

Namespaces

- namespace `magrathea`

Functions

Locate immutable element from position.

Locates the most refined cell at the provided position and returns an immutable iterator to it.

Template Parameters

Iterator	(Iterator type.)
Types	(Scalar position types.)

Parameters

in	iposs	Real positions along each dimension.
----	-------	--------------------------------------

Returns

Immutable iterator to the element found at the specified position.

- template<typename SelfType , class SelfIndex , class SelfData , unsigned int SelfDimension, class SelfPosition , class SelfExtent , class SelfElement , class SelfContainer >
`std::ostream & magrathea::operator<< (std::ostream &lhs, const SimpleHyperOctree< SelfType, SelfIndex, SelfData, SelfDimension, SelfPosition, SelfExtent, SelfElement, SelfContainer > &rhs)`
Output stream operator.

7.47.1 Detailed Description

A simple hyperoctree based on bit manipulations.

Author

Vincent Reverdy (vince.rev@gmail.com), Michel-Andr s Breton (michel-andres.breton@obspm.fr)

Date

2012-2013, 2015-2016

Copyright

CECILL-B License

7.48 /data/home/mbreton/magrathea_pathfinder/src/magrathea/simplehyperoctreeindex.h File Reference

A simple hyperoctree index based on an integer.

```
#include <iostream>
#include <iomanip>
#include <type_traits>
#include <typeinfo>
#include <limits>
#include <cstdint>
#include <string>
#include <utility>
#include <algorithm>
#include <set>
#include <array>
#include <tuple>
#include <ratio>
```

Classes

- exception [magrathea::SimpleHyperOctreeIndex< Type, Dimension, Bits >](#)
A simple hyperoctree index based on an integer.

Namespaces

- namespace [magrathea](#)

Functions

- template<typename SelfType , unsigned int SelfDimension, unsigned int SelfBits>
`std::ostream & magrathea::operator<< (std::ostream &lhs, const SimpleHyperOctreeIndex< SelfType, Self-Dimension, SelfBits > &rhs)`
Output stream operator.

7.48.1 Detailed Description

A simple hyperoctree index based on an integer.

Author

Vincent Reverdy (vince.rev@gmail.com)

Date

2012-2013

Copyright

CECILL-B License

7.49 /data/home/mbreton/magrathea_pathfinder/src/magrathea/staticvector.h File Reference

Basic vectorized constant size container.

```
#include <iostream>
#include <iomanip>
#include <sstream>
#include <type_traits>
#include <initializer_list>
#include <array>
#include "staticvectorizer.h"
```

Classes

- exception [magrathea::StaticVector< Type, Size >](#)

Basic vectorized constant size container.

Namespaces

- namespace [magrathea](#)

7.49.1 Detailed Description

Basic vectorized constant size container.

Author

Vincent Reverdy (vince.rev@gmail.com)

Date

2012-2013

Copyright

CECILL-B License

7.50 /data/home/mbreton/magrathea_pathfinder/src/magrathea/staticvectorizer.h File Reference

Helper class for generic constant size vectorization.

```
#include <iostream>
#include <iomanip>
#include <type_traits>
#include <stdexcept>
#include <functional>
#include <algorithm>
#include <iterator>
#include <initializer_list>
#include <vector>
#include <set>
#include "vectorizer.h"
```

Classes

- class [magrathea::StaticVectorizer< Kind, Size, Crtp, Type, Parameters >](#)
Helper class for generic constant size vectorization.

Namespaces

- namespace [magrathea](#)

Functions

- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, SelfType>::value)>::type>
SelfCrtp< typename
std::common_type< SelfType,
OtherType >::type,
SelfParameters...> [magrathea::operator+](#) (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> &rhs)
Addition with lhs value.
template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, SelfType>::value)>::type>
SelfCrtp< typename
std::common_type< SelfType,
OtherType >::type,
SelfParameters...> [magrathea::operator-](#) (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> &rhs)
Subtraction with lhs value.
template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, SelfType>::value)>::type>
SelfCrtp< typename
std::common_type< SelfType,
OtherType >::type,
SelfParameters...> [magrathea::operator*](#) (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> &rhs)
Multiplication with lhs value.
template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, SelfType>::value)>::type>
SelfCrtp< typename
std::common_type< SelfType,
OtherType >::type,
SelfParameters...> [magrathea::operator/](#) (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> &rhs)
Division with lhs value.
template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, SelfType>::value)>::type>
SelfCrtp< typename
std::common_type< SelfType,
OtherType >::type,
SelfParameters...> [magrathea::operator%](#) (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> &rhs)

Modulo with lhs value.

- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, SelfType>::value)>::type>
SelfCrtp< typename
std::common_type< SelfType,
OtherType >::type,
SelfParameters...> **magrathea::operator&** (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> &rhs)

Bitwise AND with lhs value.

- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, SelfType>::value)>::type>
SelfCrtp< typename
std::common_type< SelfType,
OtherType >::type,
SelfParameters...> **magrathea::operator|** (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> &rhs)

Bitwise OR with lhs value.

- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, SelfType>::value)>::type>
SelfCrtp< typename
std::common_type< SelfType,
OtherType >::type,
SelfParameters...> **magrathea::operator^** (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> &rhs)

Bitwise XOR with lhs value.

- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, SelfType>::value)>::type>
SelfCrtp< OtherType,
SelfParameters...> **magrathea::operator<<** (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> &rhs)

Bitwise left shift with lhs value.

- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, SelfType>::value)>::type>
SelfCrtp< OtherType,
SelfParameters...> **magrathea::operator>>** (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> &rhs)

Bitwise right shift with lhs value.

- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, SelfType>::value)>::type>
SelfCrtp< bool, SelfParameters...> **magrathea::operator&&** (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> &rhs)

Logical AND with lhs value.

- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, SelfType>::value)>::type>
SelfCrtp< bool, SelfParameters...> **magrathea::operator||** (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> &rhs)

Logical OR with lhs value.

- template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) &&

`(std::is_convertible<OtherType, SelfType>::value)>::type>`
`SelfCrtp< bool, SelfParameters...> magrathea::operator== (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> &rhs)`

Equal to with lhs value.

- `template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, SelfType>::value)>::type>`
`SelfCrtp< bool, SelfParameters...> magrathea::operator!= (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> &rhs)`

Not equal to with lhs value.

- `template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, SelfType>::value)>::type>`
`SelfCrtp< bool, SelfParameters...> magrathea::operator> (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> &rhs)`

Greater than with lhs value.

- `template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, SelfType>::value)>::type>`
`SelfCrtp< bool, SelfParameters...> magrathea::operator< (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> &rhs)`

Less than with lhs value.

- `template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, SelfType>::value)>::type>`
`SelfCrtp< bool, SelfParameters...> magrathea::operator>= (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> &rhs)`

Greater than or equal to with lhs value.

- `template<typename OtherType , typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters, class = typename std::enable_if<(!std::is_base_of<Vectorizer, OtherType>::value) && (std::is_convertible<OtherType, SelfType>::value)>::type>`
`SelfCrtp< bool, SelfParameters...> magrathea::operator<= (const OtherType &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> &rhs)`

Less than or equal to with lhs value.

- `template<typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters>`
`std::ostream & magrathea::operator<< (std::ostream &lhs, const StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> &rhs)`

Output stream operator.

- `template<typename SelfKind , unsigned int SelfSize, template< typename, SelfKind...> class SelfCrtp, typename SelfType , SelfKind... SelfParameters>`
`std::istream & magrathea::operator>> (std::istream &lhs, StaticVectorizer< SelfKind, SelfSize, SelfCrtp, SelfType, SelfParameters...> &rhs)`

Input stream operator.

7.50.1 Detailed Description

Helper class for generic constant size vectorization.

Author

Vincent Reverdy (vince.rev@gmail.com)

Date

2012-2013

Copyright

CECILL-B License

7.51 /data/home/mbreton/magrathea_pathfinder/src/magrathea/step.h File Reference

Basic implementation of an evolution step.

```
#include <iostream>
#include <iomanip>
#include <type_traits>
#include <typeinfo>
#include <utility>
#include <tuple>
#include <array>
#include <ratio>
#include <string>
#include <sstream>
#include "abstractstep.h"
```

Classes

- exception [magrathea::Step< Scalar, Array, Tuple >](#)

Basic implementation of an evolution step.

Namespaces

- namespace [magrathea](#)

7.51.1 Detailed Description

Basic implementation of an evolution step.

Author

Vincent Reverdy (vince.rev@gmail.com)

Date

2012-2013

Copyright

CECILL-B License

7.52 /data/home/mbreton/magrathea_pathfinder/src/magrathea/substance.h File Reference

Basic implementation of geometrical substance.

```
#include <iostream>
#include <iomanip>
#include <type_traits>
#include <typeinfo>
#include <utility>
#include <tuple>
#include <array>
#include <ratio>
#include <string>
#include <sstream>
#include "abstractsubstance.h"
```

Classes

- exception [magrathea::Substance< Types >](#)
Basic implementation of geometrical substance.

Namespaces

- namespace [magrathea](#)

7.52.1 Detailed Description

Basic implementation of geometrical substance.

Author

Vincent Reverdy (vince.rev@gmail.com)

Date

2012-2013

Copyright

CECILL-B License

7.53 /data/home/mbreton/magrathea_pathfinder/src/magrathea/temporaryfile.h File Reference

Temporary file with automatic deletion at destruction.

```
#include <iostream>
#include <iomanip>
```

Namespaces

- namespace [magrathea](#)

7.53.1 Detailed Description

Temporary file with automatic deletion at destruction.

Author

Vincent Reverdy (vince.rev@gmail.com)

Date

2012-2013

Copyright

CECILL-B License

7.54 /data/home/mbreton/magrathea_pathfinder/src/magrathea/textfile.h File Reference

Base class to manage generic text files.

```
#include <iostream>
#include <iomanip>
```

Namespaces

- namespace [magrathea](#)

7.54.1 Detailed Description

Base class to manage generic text files.

Author

Vincent Reverdy (vince.rev@gmail.com)

Date

2012-2013

Copyright

CECILL-B License

7.55 /data/home/mbreton/magrathea_pathfinder/src/magrathea/timer.h File Reference

A timer to manage time measurements and benchmarks.

```
#include <iostream>
#include <iomanip>
#include <type_traits>
#include <functional>
#include <ratio>
#include <chrono>
#include <ctime>
#include <cmath>
#include <utility>
#include "timer.h"
```

Classes

- exception [magrathea::Timer< Type, Period, Clock >](#)
A timer to manage time measurements and benchmarks.

Namespaces

- namespace [magrathea](#)

Functions

- template<typename SelfType , class SelfPeriod , class SelfClock >
`std::ostream & magrathea::operator<< (std::ostream &lhs, const Timer< SelfType, SelfPeriod, SelfClock > &rhs)`
Output stream operator.

7.55.1 Detailed Description

A timer to manage time measurements and benchmarks.

Author

Vincent Reverdy (vince.rev@gmail.com)

Date

2012-2013

Copyright

CECILL-B License

7.56 /data/home/mbreton/magrathea_pathfinder/src/magrathea/vectorized.h File Reference

Basic vectorized container.

```
#include <iostream>
#include <iomanip>
#include <initializer_list>
#include <type_traits>
#include <array>
#include "vectorizer.h"
```

Classes

- exception [magrathea::Vectorized< Type, Size >](#)

Basic vectorized container.

Namespaces

- namespace [magrathea](#)

Functions

- template<typename SelfType , unsigned int SelfSize>
`std::ostream & magrathea::operator<< (std::ostream &lhs, const Vectorized< SelfType, SelfSize > &rhs)`
Output stream operator.

7.56.1 Detailed Description

Basic vectorized container.

Author

Vincent Reverdy (vince.rev@gmail.com)

Date

2012-2013

Copyright

CECILL-B License

7.57 /data/home/mbreton/magrathea_pathfinder/src/magrathea/vectorizer.h File Reference

Helper base class for generic vectorization.

```
#include <iostream>
#include <iomanip>
#include <type_traits>
#include <stdexcept>
#include <functional>
#include <initializer_list>
#include <vector>
#include <array>
```

Classes

- class [magrathea::Vectorizer](#)
Helper base class for generic vectorization.

Namespaces

- namespace [magrathea](#)

7.57.1 Detailed Description

Helper base class for generic vectorization.

Author

Vincent Reverdy (vince.rev@gmail.com)

Date

2012-2013

Copyright

CECILL-B License

7.58 /data/home/mbreton/magrathea_pathfinder/src/magrathea(wrapper.h File Reference

Basic value wrapper with getter and setter.

```
#include <iostream>
#include <iomanip>
#include <type_traits>
#include <string>
```

Classes

- exception [magrathea::Wrapper< Type >](#)
Basic value wrapper with getter and setter.

Namespaces

- namespace [magrathea](#)

7.58.1 Detailed Description

Basic value wrapper with getter and setter.

Author

Vincent Reverdy (vince.rev@gmail.com)

Date

2012-2013

Copyright

CECILL-B License

7.59 /data/home/mbreton/magrathea_pathfinder/src/miscellaneous.h File Reference

Some miscellaneous function.

```
#include <ctime>
#include <iostream>
#include <iomanip>
#include <fstream>
#include <algorithm>
#include <utility>
#include <random>
#include <string>
#include <vector>
#include <deque>
#include <array>
#include <tuple>
#include <atomic>
#include <mutex>
#include <mpi.h>
#include "magrathea/simplehyperoctree.h"
#include "magrathea/simplehyperoctreeindex.h"
#include "magrathea/hypersphere.h"
#include "magrathea/abstracthypersphere.h"
#include "magrathea/hypercube.h"
#include "magrathea/constants.h"
#include "magrathea/evolution.h"
#include "magrathea/timer.h"
#include "cone.h"
#include "utility.h"
#include "input.h"
#include "output.h"
#include "TReadHDF5.h"
```

Classes

- class [Miscellaneous](#)

TypeDefs

- using [integer](#) = int
- using [uint](#) = unsigned int
- using [real](#) = double
- using [floating](#) = float
- using [point](#) = std::array<[real](#), 3 >

7.59.1 Detailed Description

Some miscellaneous function.

Author

Vincent Reverdy (vince.rev@gmail.com), Michel-Andr s Breton (michel-andres.breton@obspm.fr)

Date

2012-2013, 2015-2021

Copyright

CECILL-B License

7.59.2 Typedef Documentation

7.59.2.1 using floating = float

7.59.2.2 using integer = int

7.59.2.3 using point = std::array<real, 3>

7.59.2.4 using real = double

7.59.2.5 using uint = unsigned int

7.60 /data/home/mbreton/magrathea_pathfinder/src/observer_velocity.cpp File Reference

Main function of the raytracer to compute the observer's velocity.

```
#include <ctime>
#include <iostream>
#include <iomanip>
#include <fstream>
#include <algorithm>
#include <utility>
#include <random>
#include <string>
#include <vector>
#include <deque>
#include <array>
#include <tuple>
#include <atomic>
#include <mutex>
#include <mpi.h>
#include "magrathea/simplehyperoctree.h"
#include "magrathea/simplehyperoctreeindex.h"
#include "magrathea/hypersphere.h"
#include "magrathea/hypercube.h"
#include "magrathea/constants.h"
#include "magrathea/evolution.h"
#include "magrathea/timer.h"
#include "cone.h"
#include "utility.h"
#include "input.h"
#include "output.h"
#include "integrator.h"
#include "miscellaneous.h"
#include "observer_velocity.h"
#include "gravity.h"
#include "TReadHDF5.h"
```

Functions

- int **main** (int argc, char *argv[])

Main function.

7.60.1 Detailed Description

Main function of the raytracer to compute the observer's velocity.

Author

Vincent Reverdy (vince.rev@gmail.com), Michel-Andr s Breton (michel-andres.breton@obspm.fr)

Date

2012-2013, 2015-2021

Copyright

CECILL-B License

7.60.2 Function Documentation

7.60.2.1 int main (int *argc*, char * *argv*[])

Main function.

Main function of the program. It takes a namelist as a parameter and run the raytracing using MPI parallelization. To compile it, use make To run it, use: mpirun -np NTASKS ./raytracer raytracer.txt

Parameters

in	<i>argc</i>	Number of arguments.
in	<i>argv</i>	List of arguments.

Returns

Zero on success, error code otherwise.

7.61 /data/home/mbreton/magrathea_pathfinder/src/observer_velocity.h File Reference

Some observer_velocity function.

```
#include <ctime>
#include <iostream>
#include <iomanip>
#include <fstream>
#include <algorithm>
#include <utility>
#include <random>
#include <string>
#include <vector>
#include <deque>
#include <array>
#include <tuple>
#include <atomic>
#include <mutex>
#include <mpi.h>
#include "magrathea/simplehyperoctree.h"
#include "magrathea/simplehyperoctreeindex.h"
#include "magrathea/hypersphere.h"
#include "magrathea/abstracthypersphere.h"
#include "magrathea/hypercube.h"
#include "magrathea/constants.h"
#include "magrathea/evolution.h"
#include "magrathea/timer.h"
#include "cone.h"
#include "utility.h"
#include "input.h"
#include "output.h"
#include "TReadHDF5.h"
```

Classes

- struct [parameters_t](#)
- class [Observer_velocity](#)

Macros

- `#define OBSEREVR_VELOCITY_H_INCLUDED`

Variables

- struct `parameters_t` `parameters`

7.61.1 Detailed Description

Some observer_velocity function.

Author

Vincent Reverdy (vince.rev@gmail.com), Michel-Andr s Breton (michel-andres.breton@obspm.fr)

Date

2012-2013, 2015-2021

Copyright

CECILL-B License

7.61.2 Macro Definition Documentation

7.61.2.1 `#define OBSEREVR_VELOCITY_H_INCLUDED`

7.61.3 Variable Documentation

7.61.3.1 struct `parameters_t` `parameters`

7.62 /data/home/mbreton/magrathea_pathfinder/src/output.h File Reference

[Output](#) utilities for raytracing.

```
#include <iostream>
#include <iomanip>
#include <fstream>
#include <sstream>
#include <cstdio>
#include <type_traits>
#include <algorithm>
#include <limits>
#include <string>
#include <vector>
#include <array>
#include <tuple>
#include "magrathea/simplehyperoctree.h"
#include "magrathea/simplehyperoctreeindex.h"
#include "utility.h"
#include "gravity.h"
#include "photon.h"
```

Classes

- exception [Output](#)
Output utilities for raytracing.

7.62.1 Detailed Description

[Output](#) utilities for raytracing.

Author

Vincent Reverdy (vince.rev@gmail.com), Michel-Andrès Breton (michel-andres.breton@obspm.fr)

Date

2012-2013, 2015-2021

Copyright

CECILL-B License

7.63 /data/home/mbreton/magrathea_pathfinder/src/photon.h File Reference

[Photon](#) step implementation for raytracing.

```
#include <iostream>
#include <iomanip>
#include <type_traits>
#include <array>
#include <tuple>
#include <utility>
#include "magrathea/abstractstep.h"
```

Classes

- exception [Photon< Type, Dimension >](#)
Photon step implementation for raytracing.

7.63.1 Detailed Description

[Photon](#) step implementation for raytracing.

Author

Vincent Reverdy (vince.rev@gmail.com), Michel-Andrès Breton (michel-andres.breton@obspm.fr)

Date

2012-2013, 2015-2021

Copyright

CECILL-B License

7.64 /data/home/mbreton/magrathea_pathfinder/src/rays.cpp File Reference

Main function of the raytracer.

```
#include <ctime>
#include <iostream>
#include <iomanip>
#include <fstream>
#include <algorithm>
#include <utility>
#include <random>
#include <string>
#include <vector>
#include <deque>
#include <array>
#include <tuple>
#include <atomic>
#include <mutex>
#include <mpi.h>
#include "magrathea/simplehyperoctree.h"
#include "magrathea/simplehyperoctreeindex.h"
#include "magrathea/hypersphere.h"
#include "magrathea/hypercube.h"
#include "magrathea/constants.h"
#include "magrathea/evolution.h"
#include "magrathea/timer.h"
#include "cone.h"
#include "utility.h"
#include "input.h"
#include "output.h"
#include "integrator.h"
#include "miscellaneous.h"
#include "rays.h"
#include "gravity.h"
#include "TReadHDF5.h"
```

Functions

- int **main** (int argc, char *argv[])

Main function.

7.64.1 Detailed Description

Main function of the raytracer.

Author

Vincent Reverdy (vince.rev@gmail.com), Michel-Andr s Breton (michel-andres.breton@obspm.fr)

Date

2012-2013, 2015-2021

Copyright

CECILL-B License

7.64.2 Function Documentation**7.64.2.1 int main (int *argc*, char * *argv*[])**

Main function.

Main function of the program. It takes a namelist as a parameter and run the raytracing using MPI parallelization. To compile it, use make To run it, use: mpirun -np NTASKS ./raytracer raytracer.txt

Parameters

in	<i>argc</i>	Number of arguments.
in	<i>argv</i>	List of arguments.

Returns

Zero on success, error code otherwise.

7.65 /data/home/mbreton/magrathea_pathfinder/src/rays.h File Reference

Some rays function.

```
#include <ctime>
#include <iostream>
#include <iomanip>
#include <fstream>
#include <algorithm>
#include <utility>
#include <random>
#include <string>
#include <vector>
#include <deque>
#include <array>
#include <tuple>
#include <atomic>
#include <mutex>
#include <mpi.h>
#include "magrathea/simplehyperoctree.h"
#include "magrathea/simplehyperoctreeindex.h"
#include "magrathea/hypersphere.h"
#include "magrathea/abstracthypersphere.h"
#include "magrathea/hypercube.h"
#include "magrathea/constants.h"
#include "magrathea/evolution.h"
#include "magrathea/timer.h"
#include "cone.h"
#include "utility.h"
#include "input.h"
#include "output.h"
#include "TReadHDF5.h"
```

Classes

- struct [parameters_t](#)
- class [Rays](#)

Variables

- struct [parameters_t](#) [parameters](#)

7.65.1 Detailed Description

Some rays function.

Author

Vincent Reverdy (vince.rev@gmail.com), Michel-Andrès Breton (michel-andres.breton@obspm.fr)

Date

2012-2013, 2015-2021

Copyright

CECILL-B License

7.65.2 Variable Documentation

7.65.2.1 struct parameters_t parameters

7.66 /data/home/mbreton/magrathea_pathfinder/src/TReadHDF5.h File Reference

Utilities to read HDF5 files.

```
#include <iostream>
#include <string>
#include <utility>
#include <algorithm>
#include <vector>
#include <array>
#include <random>
#include <tuple>
#include <thread>
#include <errno.h>
#include <dirent.h>
#include "magrathea/constants.h"
#include "hdf5.h"
```

Classes

- class [TReadHDF5](#)

7.66.1 Detailed Description

Utilities to read HDF5 files.

Author

Michel-Andr s Breton (michel-andres.breton@obspm.fr)

Date

2015-2021

Copyright

CECILL-B License

7.67 /data/home/mbreton/magrathea_pathfinder/src/utility.h File Reference

List of utilities for raytracing.

```
#include <iostream>
#include <iomanip>
#include <type_traits>
#include <chrono>
#include <array>
#include <tuple>
#include <utility>
#include <thread>
#include <vector>
#include <algorithm>
#include "magrathea/hypercube.h"
#include "magrathea/hypersphere.h"
#include "cone.h"
```

Classes

- exception [Utility](#)

List of utilities for raytracing.

7.67.1 Detailed Description

List of utilities for raytracing.

Author

Vincent Reverdy (vince.rev@gmail.com), Michel-Andr s Breton (michel-andres.breton@obspm.fr)

Date

2012-2013, 2015-2021

Copyright

CECILL-B License