

This is a preview - [click here to buy the full publication](#)

# INTERNATIONAL STANDARD

# ISO/IEC 23270

Second edition  
2006-09-01

---

---

## Information technology — Programming languages — C#

*Technologies de l'information — Langages de programmation — C#*

Withdrawn

---

---

Reference number  
ISO/IEC 23270:2006(E)



© ISO/IEC 2006

**PDF disclaimer**

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

Withdrawn

© ISO/IEC 2006

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office  
Case postale 56 • CH-1211 Geneva 20  
Tel. + 41 22 749 01 11  
Fax + 41 22 749 09 47  
E-mail [copyright@iso.org](mailto:copyright@iso.org)  
Web [www.iso.org](http://www.iso.org)

Published in Switzerland

## Table of Contents

<b>Foreword</b> .....	<b>xv</b>
<b>Introduction</b> .....	<b>xvi</b>
<b>1. Scope</b> .....	<b>1</b>
<b>2. Conformance</b> .....	<b>3</b>
<b>3. Normative references</b> .....	<b>5</b>
<b>4. Terms and definitions</b> .....	<b>7</b>
<b>5. Notational conventions</b> .....	<b>9</b>
<b>6. Acronyms and abbreviations</b> .....	<b>11</b>
<b>7. General description</b> .....	<b>13</b>
<b>8. Language overview</b> .....	<b>15</b>
8.1 Getting started .....	15
8.2 Types.....	16
8.2.1 Predefined types.....	17
8.2.2 Conversions.....	19
8.2.3 Array types.....	20
8.2.4 Type system unification .....	22
8.3 Variables and parameters.....	22
8.4 Automatic memory management.....	25
8.5 Expressions .....	27
8.6 Statements .....	28
8.7 Classes.....	31
8.7.1 Constants.....	33
8.7.2 Fields.....	33
8.7.3 Methods.....	34
8.7.4 Properties .....	35
8.7.5 Events.....	36
8.7.6 Operators.....	37
8.7.7 Indexers.....	38
8.7.8 Instance constructors.....	39
8.7.9 Finalizers.....	40
8.7.10 Static constructors.....	40
8.7.11 Inheritance.....	41
8.7.12 Static classes .....	42
8.7.13 Partial type declarations.....	42
8.8 Structs.....	43
8.9 Interfaces .....	44
8.10 Delegates.....	45
8.11 Enums.....	46
8.12 Namespaces and assemblies.....	46
8.13 Versioning.....	48
8.14 Extern aliases.....	49
8.15 Attributes.....	51
8.16 Generics.....	52

8.16.1 Why generics?	52
8.16.2 Creating and consuming generics	53
8.16.3 Multiple type parameters	54
8.16.4 Constraints	54
8.16.5 Generic methods	56
8.17 Anonymous methods	56
8.18 Iterators	59
8.19 Nullable types	62
<b>9. Lexical structure</b>	<b>65</b>
9.1 Programs	65
9.2 Grammars	65
9.2.1 Lexical grammar	65
9.2.2 Syntactic grammar	65
9.2.3 Grammar ambiguities	66
9.3 Lexical analysis	66
9.3.1 Line terminators	67
9.3.2 Comments	67
9.3.3 White space	69
9.4 Tokens	69
9.4.1 Unicode escape sequences	69
9.4.2 Identifiers	70
9.4.3 Keywords	71
9.4.4 Literals	72
9.4.4.1 Boolean literals	72
9.4.4.2 Integer literals	72
9.4.4.3 Real literals	73
9.4.4.4 Character literals	74
9.4.4.5 String literals	75
9.4.4.6 The null literal	76
9.4.5 Operators and punctuators	77
9.5 Pre-processing directives	77
9.5.1 Conditional compilation symbols	78
9.5.2 Pre-processing expressions	78
9.5.3 Declaration directives	79
9.5.4 Conditional compilation directives	80
9.5.5 Diagnostic directives	82
9.5.6 Region control	83
9.5.7 Line directives	83
9.5.8 Pragma directives	84
<b>10. Basic concepts</b>	<b>85</b>
10.1 Application startup	85
10.2 Application termination	86
10.3 Declarations	86
10.4 Members	89
10.4.1 Namespace members	89
10.4.2 Struct members	89
10.4.3 Enumeration members	89
10.4.4 Class members	89
10.4.5 Interface members	90
10.4.6 Array members	90
10.4.7 Delegate members	90
10.5 Member access	90
10.5.1 Declared accessibility	90

10.5.2 Accessibility domains .....	91
10.5.3 Protected access for instance members .....	93
10.5.4 Accessibility constraints.....	94
10.6 Signatures and overloading .....	95
10.7 Scopes .....	96
10.7.1 Name hiding.....	98
10.7.1.1 Hiding through nesting.....	98
10.7.1.2 Hiding through inheritance.....	99
10.8 Namespace and type names.....	100
10.8.1 Unqualified name.....	102
10.8.2 Fully qualified names.....	102
10.9 Automatic memory management .....	103
10.10 Execution order .....	105
<b>11. Types .....</b>	<b>107</b>
11.1 Value types.....	107
11.1.1 The System.ValueType type.....	108
11.1.2 Default constructors .....	108
11.1.3 Struct types.....	109
11.1.4 Simple types.....	109
11.1.5 Integral types.....	110
11.1.6 Floating point types.....	111
11.1.7 The decimal type.....	111
11.1.8 The bool type .....	112
11.1.9 Enumeration types.....	112
11.2 Reference types .....	112
11.2.1 Class types.....	113
11.2.2 The object type.....	113
11.2.3 The string type .....	113
11.2.4 Interface types.....	113
11.2.5 Array types.....	114
11.2.6 Delegate types.....	114
11.2.7 The null type.....	114
11.3 Boxing and unboxing.....	114
11.3.1 Boxing conversions.....	114
11.3.2 Unboxing conversions.....	115
11.4 Nullable types.....	116
11.4.1 Members.....	116
11.4.2 Implemented interfaces .....	117
<b>12. Variables .....</b>	<b>119</b>
12.1 Variable categories.....	119
12.1.1 Static variables .....	119
12.1.2 Instance variables.....	119
12.1.2.1 Instance variables in classes.....	119
12.1.2.2 Instance variables in structs.....	120
12.1.3 Array elements .....	120
12.1.4 Value parameters.....	120
12.1.5 Reference parameters.....	120
12.1.6 Output parameters .....	120
12.1.7 Local variables .....	121
12.2 Default values.....	121
12.3 Definite assignment.....	122
12.3.1 Initially assigned variables.....	123
12.3.2 Initially unassigned variables.....	123

12.3.3 Precise rules for determining definite assignment .....	123
12.3.3.1 General rules for statements .....	124
12.3.3.2 Block statements, checked, and unchecked statements .....	124
12.3.3.3 Expression statements .....	124
12.3.3.4 Declaration statements .....	124
12.3.3.5 If statements .....	124
12.3.3.6 Switch statements .....	125
12.3.3.7 While statements .....	125
12.3.3.8 Do statements .....	125
12.3.3.9 For statements .....	125
12.3.3.10 Break, continue, and goto statements .....	126
12.3.3.11 Throw statements .....	126
12.3.3.12 Return statements .....	126
12.3.3.13 Try-catch statements .....	126
12.3.3.14 Try-finally statements .....	127
12.3.3.15 Try-catch-finally statements .....	127
12.3.3.16 Foreach statements .....	128
12.3.3.17 Using statements .....	128
12.3.3.18 Lock statements .....	128
12.3.3.19 General rules for simple expressions .....	128
12.3.3.20 General rules for expressions with embedded expressions .....	129
12.3.3.21 Invocation expressions and object creation expressions .....	129
12.3.3.22 Simple assignment expressions .....	129
12.3.3.23 && expressions .....	130
12.3.3.24    expressions .....	131
12.3.3.25 ! expressions .....	131
12.3.3.26 ?: expressions .....	132
12.3.3.27 Anonymous method expressions .....	132
12.3.3.28 Yield statements .....	133
12.3.3.29 ?? expressions .....	133
12.4 Variable references .....	133
12.5 Atomicity of variable references .....	133
<b>13. Conversions.....</b>	<b>135</b>
13.1 Implicit conversions .....	135
13.1.1 Identity conversion .....	135
13.1.2 Implicit numeric conversions .....	135
13.1.3 Implicit enumeration conversions .....	136
13.1.4 Implicit reference conversions .....	136
13.1.5 Boxing conversions .....	137
13.1.6 Implicit type parameter conversions .....	137
13.1.7 Implicit constant expression conversions .....	138
13.1.8 User-defined implicit conversions .....	138
13.2 Explicit conversions .....	138
13.2.1 Explicit numeric conversions .....	138
13.2.2 Explicit enumeration conversions .....	140
13.2.3 Explicit reference conversions .....	140
13.2.4 Unboxing conversions .....	141
13.2.5 Explicit type parameter conversions .....	141
13.2.6 User-defined explicit conversions .....	142
13.3 Standard conversions .....	142
13.3.1 Standard implicit conversions .....	142
13.3.2 Standard explicit conversions .....	142
13.4 User-defined conversions .....	142
13.4.1 Permitted user-defined conversions .....	142

13.4.2 Evaluation of user-defined conversions .....	143
13.4.3 User-defined implicit conversions .....	144
13.4.4 User-defined explicit conversions .....	144
13.5 Anonymous method conversions .....	145
13.6 Method group conversions .....	146
13.7 Conversions involving nullable types .....	147
13.7.1 Null type conversions .....	148
13.7.2 Nullable conversions .....	148
13.7.3 Lifted conversions .....	148
<b>14. Expressions .....</b>	<b>149</b>
14.1 Expression classifications .....	149
14.1.1 Values of expressions .....	150
14.2 Operators .....	150
14.2.1 Operator precedence and associativity .....	150
14.2.2 Operator overloading .....	151
14.2.3 Unary operator overload resolution .....	152
14.2.4 Binary operator overload resolution .....	153
14.2.5 Candidate user-defined operators .....	153
14.2.6 Numeric promotions .....	153
14.2.6.1 Unary numeric promotions .....	154
14.2.6.2 Binary numeric promotions .....	154
14.2.7 Lifted operators .....	155
14.3 Member lookup .....	156
14.3.1 Base types .....	157
14.4 Function members .....	157
14.4.1 Argument lists .....	159
14.4.2 Overload resolution .....	161
14.4.2.1 Applicable function member .....	162
14.4.2.2 Better function member .....	163
14.4.2.3 Better conversion .....	163
14.4.3 Function member invocation .....	164
14.4.3.1 Invocations on boxed instances .....	165
14.5 Primary expressions .....	165
14.5.1 Literals .....	166
14.5.2 Simple names .....	166
14.5.2.1 Invariant meaning in blocks .....	168
14.5.3 Parenthesized expressions .....	168
14.5.4 Member access .....	169
14.5.4.1 Identical simple names and type names .....	170
14.5.5 Invocation expressions .....	171
14.5.5.1 Method invocations .....	171
14.5.5.2 Delegate invocations .....	172
14.5.6 Element access .....	173
14.5.6.1 Array access .....	173
14.5.6.2 Indexer access .....	173
14.5.7 This access .....	174
14.5.8 Base access .....	175
14.5.9 Postfix increment and decrement operators .....	175
14.5.10 The new operator .....	176
14.5.10.1 Object creation expressions .....	176
14.5.10.2 Array creation expressions .....	178
14.5.10.3 Delegate creation expressions .....	179
14.5.11 The typeof operator .....	182
14.5.12 The sizeof operator .....	184

14.5.13	The checked and unchecked operators.....	184
14.5.14	Default value expression.....	187
14.5.15	Anonymous methods.....	187
14.5.15.1	Anonymous method signatures.....	187
14.5.15.2	Anonymous method blocks.....	188
14.5.15.3	Outer variables.....	188
14.5.15.4	Anonymous method evaluation.....	191
14.5.15.5	Implementation example.....	191
14.6	Unary expressions.....	194
14.6.1	Unary plus operator.....	194
14.6.2	Unary minus operator.....	194
14.6.3	Logical negation operator.....	195
14.6.4	Bitwise complement operator.....	195
14.6.5	Prefix increment and decrement operators.....	195
14.6.6	Cast expressions.....	196
14.7	Arithmetic operators.....	197
14.7.1	Multiplication operator.....	197
14.7.2	Division operator.....	198
14.7.3	Remainder operator.....	199
14.7.4	Addition operator.....	200
14.7.5	Subtraction operator.....	202
14.8	Shift operators.....	204
14.9	Relational and type-testing operators.....	205
14.9.1	Integer comparison operators.....	206
14.9.2	Floating-point comparison operators.....	207
14.9.3	Decimal comparison operators.....	207
14.9.4	Boolean equality operators.....	208
14.9.5	Enumeration comparison operators.....	208
14.9.6	Reference type equality operators.....	208
14.9.7	String equality operators.....	210
14.9.8	Delegate equality operators.....	210
14.9.9	Equality operators and null.....	211
14.9.10	is operator.....	211
14.9.11	as operator.....	212
14.10	Logical operators.....	213
14.10.1	Integer logical operators.....	213
14.10.2	Enumeration logical operators.....	214
14.10.3	Boolean logical operators.....	214
14.10.4	The bool? logical operators.....	214
14.11	Conditional logical operators.....	215
14.11.1	Boolean conditional logical operators.....	215
14.11.2	User-defined conditional logical operators.....	216
14.12	The null coalescing operator.....	216
14.13	Conditional operator.....	217
14.14	Assignment operators.....	218
14.14.1	Simple assignment.....	218
14.14.2	Compound assignment.....	220
14.14.3	Event assignment.....	221
14.15	Expression.....	221
14.16	Constant expressions.....	221
14.17	Boolean expressions.....	222
<b>15.</b>	<b>Statements.....</b>	<b>225</b>
15.1	End points and reachability.....	225
15.2	Blocks.....	227



15.2.1 Statement lists .....	227
15.3 The empty statement .....	227
15.4 Labeled statements .....	228
15.5 Declaration statements .....	228
15.5.1 Local variable declarations .....	229
15.5.2 Local constant declarations .....	229
15.6 Expression statements .....	230
15.7 Selection statements .....	230
15.7.1 The if statement .....	230
15.7.2 The switch statement .....	231
15.8 Iteration statements .....	234
15.8.1 The while statement .....	234
15.8.2 The do statement .....	235
15.8.3 The for statement .....	235
15.8.4 The foreach statement .....	236
15.9 Jump statements .....	239
15.9.1 The break statement .....	240
15.9.2 The continue statement .....	241
15.9.3 The goto statement .....	241
15.9.4 The return statement .....	242
15.9.5 The throw statement .....	243
15.10 The try statement .....	244
15.11 The checked and unchecked statements .....	246
15.12 The lock statement .....	247
15.13 The using statement .....	247
15.14 The yield statement .....	249
<b>16. Namespaces .....</b>	<b>251</b>
16.1 Compilation units .....	251
16.2 Namespace declarations .....	251
16.3 Extern alias directives .....	252
16.4 Using directives .....	253
16.4.1 Using alias directives .....	253
16.4.2 Using namespace directives .....	257
16.5 Namespace members .....	259
16.6 Type declarations .....	259
16.7 Qualified alias member .....	259
<b>17. Classes .....</b>	<b>263</b>
17.1 Class declarations .....	263
17.1.1 Class modifiers .....	263
17.1.1.1 Abstract classes .....	264
17.1.1.2 Sealed classes .....	264
17.1.1.3 Static classes .....	264
17.1.2 Class base specification .....	265
17.1.2.1 Base classes .....	266
17.1.2.2 Interface implementations .....	267
17.1.3 Class body .....	268
17.1.4 Partial declarations .....	268
17.2 Class members .....	269
17.2.1 Inheritance .....	271
17.2.2 The new modifier .....	272
17.2.3 Access modifiers .....	272
17.2.4 Constituent types .....	272
17.2.5 Static and instance members .....	272

17.2.6 Nested types .....	273
17.2.6.1 Fully qualified name .....	273
17.2.6.2 Declared accessibility .....	274
17.2.6.3 Hiding .....	274
17.2.6.4 this access .....	275
17.2.6.5 Access to private and protected members of the containing type .....	275
17.2.7 Reserved member names .....	276
17.2.7.1 Member names reserved for properties .....	276
17.2.7.2 Member names reserved for events .....	277
17.2.7.3 Member names reserved for indexers .....	277
17.2.7.4 Member names reserved for finalizers .....	277
17.3 Constants .....	277
17.4 Fields .....	279
17.4.1 Static and instance fields .....	280
17.4.2 Readonly fields .....	280
17.4.2.1 Using static readonly fields for constants .....	281
17.4.2.2 Versioning of constants and static readonly fields .....	281
17.4.3 Volatile fields .....	282
17.4.4 Field initialization .....	283
17.4.5 Variable initializers .....	283
17.4.5.1 Static field initialization .....	284
17.4.5.2 Instance field initialization .....	285
17.5 Methods .....	285
17.5.1 Method parameters .....	287
17.5.1.1 Value parameters .....	288
17.5.1.2 Reference parameters .....	288
17.5.1.3 Output parameters .....	289
17.5.1.4 Parameter arrays .....	290
17.5.2 Static and instance methods .....	292
17.5.3 Virtual methods .....	292
17.5.4 Override methods .....	294
17.5.5 Sealed methods .....	296
17.5.6 Abstract methods .....	296
17.5.7 External methods .....	297
17.5.8 Method body .....	298
17.5.9 Method overloading .....	299
17.6 Properties .....	299
17.6.1 Static and instance properties .....	300
17.6.2 Accessors .....	300
17.6.3 Virtual, sealed, override, and abstract accessors .....	306
17.7 Events .....	307
17.7.1 Field-like events .....	309
17.7.2 Event accessors .....	312
17.7.3 Static and instance events .....	313
17.7.4 Virtual, sealed, override, and abstract accessors .....	313
17.8 Indexers .....	314
17.8.1 Indexer overloading .....	317
17.9 Operators .....	317
17.9.1 Unary operators .....	318
17.9.2 Binary operators .....	319
17.9.3 Conversion operators .....	320
17.10 Instance constructors .....	321
17.10.1 Constructor initializers .....	322
17.10.2 Instance variable initializers .....	322
17.10.3 Constructor execution .....	323

17.10.4 Default constructors .....	324
17.10.5 Private constructors .....	325
17.10.6 Optional instance constructor parameters .....	325
17.11 Static constructors .....	326
17.12 Finalizers .....	327
<b>18. Structs .....</b>	<b>331</b>
18.1 Struct declarations .....	331
18.1.1 Struct modifiers .....	331
18.1.2 Struct interfaces .....	332
18.1.3 Struct body .....	332
18.2 Struct members .....	332
18.3 Class and struct differences .....	332
18.3.1 Value semantics .....	332
18.3.2 Inheritance .....	333
18.3.3 Assignment .....	333
18.3.4 Default values .....	333
18.3.5 Boxing and unboxing .....	334
18.3.6 Meaning of this .....	334
18.3.7 Field initializers .....	334
18.3.8 Constructors .....	335
18.3.9 Finalizers .....	335
18.3.10 Static constructors .....	335
<b>19. Arrays .....</b>	<b>337</b>
19.1 Array types .....	337
19.1.1 The System.Array type .....	338
19.2 Array creation .....	338
19.3 Array element access .....	338
19.4 Array members .....	338
19.5 Array covariance .....	338
19.6 Arrays and the generic IList interface .....	339
19.7 Array initializers .....	340
<b>20. Interfaces .....</b>	<b>343</b>
20.1 Interface declarations .....	343
20.1.1 Interface modifiers .....	343
20.1.2 Base interfaces .....	344
20.1.3 Interface body .....	344
20.2 Interface members .....	345
20.2.1 Interface methods .....	346
20.2.2 Interface properties .....	346
20.2.3 Interface events .....	346
20.2.4 Interface indexers .....	346
20.2.5 Interface member access .....	347
20.3 Fully qualified interface member names .....	348
20.4 Interface implementations .....	349
20.4.1 Explicit interface member implementations .....	349
20.4.2 Interface mapping .....	351
20.4.3 Interface implementation inheritance .....	354
20.4.4 Interface re-implementation .....	356
20.4.5 Abstract classes and interfaces .....	357
<b>21. Enums .....</b>	<b>359</b>
21.1 Enum declarations .....	359
21.2 Enum modifiers .....	359

21.3 Enum members.....	360
21.4 The System.Enum type .....	362
21.5 Enum values and operations.....	362
<b>22. Delegates.....</b>	<b>363</b>
22.1 Delegate declarations .....	363
22.2 Delegate instantiation.....	365
22.3 Delegate invocation.....	365
<b>23. Exceptions .....</b>	<b>369</b>
23.1 Causes of exceptions .....	369
23.2 The System.Exception class.....	369
23.3 How exceptions are handled.....	369
23.4 Common exception classes .....	370
<b>24. Attributes .....</b>	<b>373</b>
24.1 Attribute classes .....	373
24.1.1 Attribute usage .....	373
24.1.2 Positional and named parameters.....	374
24.1.3 Attribute parameter types.....	375
24.2 Attribute specification.....	375
24.3 Attribute instances.....	380
24.3.1 Compilation of an attribute .....	380
24.3.2 Run-time retrieval of an attribute instance.....	381
24.4 Reserved attributes.....	381
24.4.1 The AttributeUsage attribute.....	382
24.4.2 The Conditional attribute .....	382
24.4.2.1 Conditional methods .....	382
24.4.2.2 Conditional attribute classes.....	384
24.4.3 The Obsolete attribute.....	385
<b>25. Generics.....</b>	<b>387</b>
25.1 Generic class declarations .....	387
25.1.1 Type parameters.....	387
25.1.2 The instance type.....	388
25.1.3 Members of generic classes .....	389
25.1.4 Static fields in generic classes.....	389
25.1.5 Static constructors in generic classes .....	390
25.1.6 Accessing protected members.....	390
25.1.7 Overloading in generic classes.....	391
25.1.8 Parameter array methods and type parameters.....	391
25.1.9 Overriding and generic classes.....	392
25.1.10 Operators in generic classes.....	392
25.1.11 Nested types in generic classes .....	393
25.2 Generic struct declarations .....	394
25.3 Generic interface declarations.....	394
25.3.1 Uniqueness of implemented interfaces .....	395
25.3.2 Explicit interface member implementations .....	396
25.4 Generic delegate declarations.....	396
25.5 Constructed types .....	397
25.5.1 Type arguments.....	397
25.5.2 Open and closed types.....	398
25.5.3 Base classes and interfaces of a constructed type .....	398
25.5.4 Members of a constructed type .....	399
25.5.5 Accessibility of a constructed type .....	399
25.5.6 Conversions.....	400

25.5.7 Using alias directives .....	400
25.6 Generic methods.....	400
25.6.1 Generic method signatures.....	401
25.6.2 Virtual generic methods.....	401
25.6.3 Calling generic methods.....	403
25.6.4 Inference of type arguments.....	403
25.6.5 Using a generic method with a delegate.....	405
25.6.6 No generic properties, events, indexers, operators, constructors, or finalizers.....	405
25.7 Constraints.....	405
25.7.1 Satisfying constraints .....	410
25.7.2 Member lookup on type parameters.....	410
25.7.3 Type parameters and boxing .....	411
25.7.4 Conversions involving type parameters .....	412
<b>26. Iterators.....</b>	<b>415</b>
26.1 Iterator blocks.....	415
26.1.1 Enumerator interfaces .....	416
26.1.2 Enumerable interfaces.....	416
26.1.3 Yield type.....	416
26.1.4 This access .....	416
26.2 Enumerator objects.....	416
26.2.1 The MoveNext method .....	417
26.2.2 The Current property.....	418
26.2.3 The Dispose method.....	418
26.3 Enumerable objects .....	418
26.3.1 The GetEnumerator method.....	419
26.4 Implementation example .....	419
<b>27. Unsafe code .....</b>	<b>425</b>
27.1 Unsafe contexts .....	425
27.2 Pointer types.....	427
27.3 Fixed and moveable variables.....	430
27.4 Pointer conversions .....	430
27.5 Pointers in expressions.....	431
27.5.1 Pointer indirection.....	432
27.5.2 Pointer member access.....	432
27.5.3 Pointer element access.....	433
27.5.4 The address-of operator.....	434
27.5.5 Pointer increment and decrement.....	435
27.5.6 Pointer arithmetic.....	435
27.5.7 Pointer comparison .....	436
27.5.8 The sizeof operator.....	436
27.6 The fixed statement .....	436
27.7 Stack allocation .....	439
27.8 Dynamic memory allocation .....	440
<b>Annex A. Grammar .....</b>	<b>443</b>
A.1 Lexical grammar.....	443
A.1.1 Line terminators .....	443
A.1.2 White space .....	443
A.1.3 Comments .....	444
A.1.4 Tokens.....	444
A.1.5 Unicode escape sequences .....	445
A.1.6 Identifiers .....	445
A.1.7 Keywords .....	446
A.1.8 Literals.....	446

A.1.9 Operators and punctuators.....	448
A.1.10 Pre-processing directives.....	448
A.2 Syntactic grammar.....	450
A.2.1 Basic concepts.....	450
A.2.2 Types.....	451
A.2.3 Variables.....	452
A.2.4 Expressions.....	452
A.2.5 Statements.....	456
A.2.6 Classes.....	460
A.2.7 Structs.....	465
A.2.8 Arrays.....	466
A.2.9 Interfaces.....	467
A.2.10 Enums.....	468
A.2.11 Delegates.....	468
A.2.12 Attributes.....	468
A.2.13 Generics.....	470
A.3 Grammar extensions for unsafe code.....	470
<b>Annex B. Portability issues.....</b>	<b>473</b>
B.1 Undefined behavior.....	473
B.2 Implementation-defined behavior.....	473
B.3 Unspecified behavior.....	474
B.4 Other Issues.....	474
<b>Annex C. Naming guidelines.....</b>	<b>475</b>
<b>Annex D. Standard library.....</b>	<b>477</b>
<b>Annex E. Documentation comments.....</b>	<b>487</b>
E.1 Introduction.....	487
E.2 Recommended tags.....	488
E.2.1 <c>.....	489
E.2.2 <code>.....	489
E.2.3 <example>.....	490
E.2.4 <exception>.....	490
E.2.5 <list>.....	490
E.2.6 <para>.....	491
E.2.7 <param>.....	492
E.2.8 <paramref>.....	492
E.2.9 <permission>.....	493
E.2.10 <remarks>.....	493
E.2.11 <returns>.....	493
E.2.12 <see>.....	494
E.2.13 <seealso>.....	494
E.2.14 <summary>.....	495
E.2.15 <typeparam>.....	495
E.2.16 <typeparamref>.....	495
E.2.17 <value>.....	496
E.3 Processing the documentation file.....	496
E.3.1 ID string format.....	496
E.3.2 ID string examples.....	497
E.4 An example.....	501
E.4.1 C# source code.....	501
E.4.2 Resulting XML.....	503
<b>Bibliography.....</b>	<b>507</b>
<b>Index.....</b>	<b>508</b>

## Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 23270 was prepared by Ecma (as Ecma-334) and was adopted, under a special “fast-track procedure”, by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, in parallel with its approval by national bodies of ISO and IEC. This second edition cancels and replaces the first edition (ISO/IEC 23270:2003), which has been technically revised.



# Introduction

This International Standard is based on a submission from Hewlett-Packard, Intel and Microsoft, that described a language called C#, which was developed within Microsoft. The principal inventors of this language were Anders Hejlsberg, Scott Wiltamuth and Peter Golde. The first widely distributed implementation of C# was released by Microsoft in July 2000, as part of its .NET Framework initiative.

Ecma Technical Committee 39 (TC39) Task Group 2 (TG2) was formed in September 2000, to produce a standard for C#. Another Task Group, TG3, was also formed at that time to produce a standard for a library and execution environment called Common Language Infrastructure (CLI). (CLI is based on a subset of the .NET Framework.) Although Microsoft's implementation of C# relies on CLI for library and runtime support, other implementations of C# need not, provided they support an alternate way of getting at the minimum CLI features required by this C# standard (see Annex D).

As the definition of C# evolved, the goals used in its design were as follows:

- C# is intended to be a simple, modern, general-purpose, object-oriented programming language.
- The language and implementations thereof should provide support for software engineering principles such as strong type checking, array bounds checking, detection of attempts to use uninitialized variables, and automatic garbage collection. Software robustness, durability and programmer productivity are important.
- The language is intended for use in developing software components suitable for deployment in distributed environments.
- Source code portability is very important, as is programmer portability, especially for those programmers already familiar with C and C++.
- Support for internationalization is very important.
- C# is intended to be suitable for writing applications for both hosted and embedded systems, ranging from the very large that use sophisticated operating systems, down to the very small having dedicated functions.
- Although C# applications are intended to be economical with regard to memory and processing power requirements, the language was not intended to compete directly on performance and size with C or assembly language.

The following companies and organizations have participated in the development of this International Standard, and their contributions are gratefully acknowledged: ActiveState, Borland, CSK Corp., Hewlett-Packard, IBM, Intel, IT University of Copenhagen, Jagersoft (UK), Microsoft, Mountain View Compiler, Monash University (AUS), Netscape, Novell, Pixo, Plum Hall, Sun, and the University of Canterbury (NZ).

The development of this version of the International Standard started in January 2003.



# Information technology — Programming languages — C#

## 1. Scope

This International Standard specifies the form and establishes the interpretation of programs written in the C# programming language. It specifies

- The representation of C# programs;
- The syntax and constraints of the C# language;
- The semantic rules for interpreting C# programs;
- The restrictions and limits imposed by a conforming implementation of C#.

This International Standard does not specify

- The mechanism by which C# programs are transformed for use by a data-processing system;
- The mechanism by which C# applications are invoked for use by a data-processing system;
- The mechanism by which input data are transformed for use by a C# application;
- The mechanism by which output data are transformed after being produced by a C# application;
- The size or complexity of a program and its data that will exceed the capacity of any specific data-processing system or the capacity of a particular processor;
- All minimal requirements of a data-processing system that is capable of supporting a conforming implementation.

## 2. Conformance

Conformance is of interest to the following audiences:

- Those designing, implementing, or maintaining C# implementations.
- Governmental or commercial entities wishing to procure C# implementations.
- Testing organizations wishing to provide a C# conformance test suite.
- Programmers wishing to port code from one C# implementation to another.
- Educators wishing to teach Standard C#.
- Authors wanting to write about Standard C#.

As such, conformance is most important, and the bulk of this International Standard is aimed at specifying the characteristics that make C# implementations and C# programs conforming ones.

The text in this International Standard that specifies requirements is considered *normative*. All other text in this specification is *informative*; that is, for information purposes only. Unless stated otherwise, all text is normative. Normative text is further broken into *required* and *conditional* categories. *Conditionally normative* text specifies a feature and its requirements where the feature is optional. However, if that feature is provided, its syntax and semantics must be exactly as specified.

Undefined behavior is indicated in this International Standard only by the words “undefined behavior.”

A *strictly conforming program* shall use only those features of the language specified in this International Standard as being required. (This means that a strictly conforming program cannot use any conditionally normative feature.) It shall not produce output dependent on any unspecified, undefined or implementation-defined behavior.

A *conforming implementation* of C# must accept any strictly conforming program.

A conforming implementation of C# must provide and support all the types, values, objects, properties, methods, and program syntax and semantics described in the normative (but not the conditionally normative) parts in this International Standard.

A conforming implementation of C# shall interpret characters in conformance with the Unicode Standard Version 4.0 and ISO/IEC 10646-1. Conforming implementations must accept Unicode source files encoded with the UTF-8 encoding form.

A conforming implementation of C# shall not successfully translate source containing a `#error` preprocessing directive unless it is part of a group skipped by conditional compilation.

A conforming implementation of C# shall produce at least one diagnostic message if the source program violates any rule of syntax or any negative requirement (defined as a “shall” or “shall not” or “error” or “warning” requirement), unless that requirement is marked with the words “no diagnostic is required”.

A conforming implementation of C# is permitted to provide additional types, values, objects, properties and methods beyond those described in this International Standard, provided they do not alter the behavior of any strictly conforming program. Conforming implementations are required to diagnose programs that use extensions that are ill formed according to this International Standard. Having done so, however; they can compile and execute such programs. (The ability to have extensions implies that a conforming implementation reserves no identifiers other than those explicitly reserved in this International Standard.)

A conforming implementation of C# shall be accompanied by a document that defines all implementation-defined characteristics and all extensions.

A conforming implementation of C# shall support the class library documented in Annex D. This library is included by reference in this International Standard.

A **conforming program** is one that is acceptable to a conforming implementation. (Such a program is permitted to contain extensions or conditionally normative features.)

Withdrawn

### 3. Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 31.11:1992, *Quantities and units — Part 11: Mathematical signs and symbols for use in the physical sciences and technology*.

ISO/IEC 2382.1:1993, *Information technology — Vocabulary — Part 1: Fundamental terms*.

ISO/IEC 10646 (all parts), *Information technology — Universal Multiple-Octet Coded Character Set (UCS)*.

ISO/IEC 23271:2005, *Common Language Infrastructure (CLI), Partition IV: Base Class Library (BCL), Extended Numerics Library, and Extended Array Library*.

IEC 60559:1989, *Binary floating-point arithmetic for microprocessor systems* (previously designated IEC 559:1989). (This standard is widely known by its U.S. national designation, ANSI/IEEE Standard 754-1985, *IEEE Standard for Binary Floating-Point Arithmetic*.)

*The Unicode Standard, Version 4.0*, The Unicode Consortium, (Addison-Wesley, Boston, MA, 2003. ISBN 0-321-18578-1).

Withhold & Copyright