
**Information technology — C# Language
Specification**

Technologies de l'information — Spécification du langage C#

iTeh STANDARD PREVIEW
(standards.iteh.ai)

[ISO/IEC 23270:2003](https://standards.iteh.ai/catalog/standards/sist/b96b32fb-28fe-4204-89dd-7eab24f88245/iso-iec-23270-2003)

<https://standards.iteh.ai/catalog/standards/sist/b96b32fb-28fe-4204-89dd-7eab24f88245/iso-iec-23270-2003>

Reference number
ISO/IEC 23270:2003(E)



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.

iTeh STANDARD PREVIEW
(standards.iteh.ai)

[ISO/IEC 23270:2003](https://standards.iteh.ai/catalog/standards/sist/b96B2fb-28fe-4204-89dd-7eab24f88245/iso-iec-23270-2003)

<https://standards.iteh.ai/catalog/standards/sist/b96B2fb-28fe-4204-89dd-7eab24f88245/iso-iec-23270-2003>

© ISO/IEC 2003

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
Web www.iso.org

Published in Switzerland

Table of Contents

1. Scope	1
2. Conformance	3
3. References	5
4. Definitions	7
5. Notational conventions	9
6. Acronyms and abbreviations	11
7. General description	13
8. Language Overview	15
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.....	21
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.....	32
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 Destructors	39
8.7.10 Static constructors.....	40
8.7.11 Inheritance	40
8.8 Structs	41
8.9 Interfaces	42
8.10 Delegates	43
8.11 Enums	44
8.12 Namespaces and assemblies	45
8.13 Versioning	46
8.14 Attributes	48
9. Lexical structure	51
9.1 Programs.....	51
9.2 Grammars	51
9.2.1 Lexical grammar	51
9.2.2 Syntactic grammar	51
9.3 Lexical analysis	52
9.3.1 Line terminators.....	52
9.3.2 Comments	53

9.3.3 White space	54
9.4 Tokens	54
9.4.1 Unicode escape sequences	54
9.4.2 Identifiers	55
9.4.3 Keywords	56
9.4.4 Literals	57
9.4.5 Operators and punctuators	62
9.5 Pre-processing directives	62
9.5.1 Conditional compilation symbols	63
9.5.2 Pre-processing expressions	63
9.5.3 Declaration directives	64
9.5.4 Conditional compilation directives	65
9.5.5 Diagnostic directives	67
9.5.6 Region control	67
9.5.7 Line directives	68
10. Basic concepts	69
10.1 Application startup	69
10.2 Application termination	69
10.3 Declarations	70
10.4 Members	72
10.4.1 Namespace members	72
10.4.2 Struct members	72
10.4.3 Enumeration members	73
10.4.4 Class members	73
10.4.5 Interface members	73
10.4.6 Array members	73
10.4.7 Delegate members	73
10.5 Member access	73
10.5.1 Declared accessibility	74
10.5.2 Accessibility domains	74
10.5.3 Protected access for instance members	77
10.5.4 Accessibility constraints	77
10.6 Signatures and overloading	78
10.7 Scopes	79
10.7.1 Name hiding	81
10.8 Namespace and type names	83
10.8.1 Fully qualified names	84
10.9 Automatic memory management	85
10.10 Execution order	87
11. Types	89
11.1 Value types	89
11.1.1 Default constructors	90
11.1.2 Struct types	90
11.1.3 Simple types	91
11.1.4 Integral types	91
11.1.5 Floating point types	92
11.1.6 The <code>decimal</code> type	94
11.1.7 The <code>bool</code> type	94
11.1.8 Enumeration types	94
11.2 Reference types	94
11.2.1 Class types	95
11.2.2 The <code>object</code> type	95
11.2.3 The <code>string</code> type	95

11.2.4 Interface types.....	96
11.2.5 Array types.....	96
11.2.6 Delegate types.....	96
11.3 Boxing and unboxing	96
11.3.1 Boxing conversions.....	96
11.3.2 Unboxing conversions	97
12. Variables	99
12.1 Variable categories	99
12.1.1 Static variables.....	99
12.1.2 Instance variables.....	99
12.1.3 Array elements.....	100
12.1.4 Value parameters	100
12.1.5 Reference parameters.....	100
12.1.6 Output parameters.....	100
12.1.7 Local variables.....	101
12.2 Default values.....	101
12.3 Definite assignment.....	102
12.3.1 Initially assigned variables.....	102
12.3.2 Initially unassigned variables.....	103
12.3.3 Precise rules for determining definite assignment	103
12.4 Variable references.....	112
12.5 Atomicity of variable references	112
13. Conversions.....	113
13.1 Implicit conversions	113
13.1.1 Identity conversion	113
13.1.2 Implicit numeric conversions.....	113
13.1.3 Implicit enumeration conversions.....	114
13.1.4 Implicit reference conversions.....	114
13.1.5 Boxing conversions.....	114
13.1.6 Implicit constant expression conversions.....	114
13.1.7 User-defined implicit conversions	115
13.2 Explicit conversions	115
13.2.1 Explicit numeric conversions.....	115
13.2.2 Explicit enumeration conversions.....	117
13.2.3 Explicit reference conversions	117
13.2.4 Unboxing conversions	117
13.2.5 User-defined explicit conversions.....	118
13.3 Standard conversions.....	118
13.3.1 Standard implicit conversions.....	118
13.3.2 Standard explicit conversions	118
13.4 User-defined conversions	118
13.4.1 Permitted user-defined conversions.....	118
13.4.2 Evaluation of user-defined conversions.....	119
13.4.3 User-defined implicit conversions	119
13.4.4 User-defined explicit conversions.....	120
14. Expressions	123
14.1 Expression classifications.....	123
14.1.1 Values of expressions	124
14.2 Operators	124
14.2.1 Operator precedence and associativity.....	124
14.2.2 Operator overloading	125
14.2.3 Unary operator overload resolution	126
14.2.4 Binary operator overload resolution	127

14.2.5 Candidate user-defined operators	127
14.2.6 Numeric promotions	127
14.3 Member lookup	129
14.3.1 Base types	129
14.4 Function members	130
14.4.1 Argument lists.....	132
14.4.2 Overload resolution.....	134
14.4.3 Function member invocation	136
14.5 Primary expressions.....	137
14.5.1 Literals	138
14.5.2 Simple names	138
14.5.3 Parenthesized expressions.....	139
14.5.4 Member access.....	140
14.5.5 Invocation expressions.....	141
14.5.6 Element access.....	143
14.5.7 This access	144
14.5.8 Base access	145
14.5.9 Postfix increment and decrement operators.....	145
14.5.10 The new operator.....	146
14.5.11 The typeof operator.....	150
14.5.12 The checked and unchecked operators	151
14.6 Unary expressions	153
14.6.1 Unary plus operator.....	153
14.6.2 Unary minus operator	154
14.6.3 Logical negation operator	154
14.6.4 Bitwise complement operator	154
14.6.5 Prefix increment and decrement operators.....	155
14.6.6 Cast expressions	155
14.7 Arithmetic operators	156
14.7.1 Multiplication operator	156
14.7.2 Division operator	157
14.7.3 Remainder operator.....	158
14.7.4 Addition operator.....	159
14.7.5 Subtraction operator.....	161
14.8 Shift operators	162
14.9 Relational and type-testing operators	163
14.9.1 Integer comparison operators.....	164
14.9.2 Floating-point comparison operators	165
14.9.3 Decimal comparison operators	165
14.9.4 Boolean equality operators	165
14.9.5 Enumeration comparison operators	166
14.9.6 Reference type equality operators.....	166
14.9.7 String equality operators.....	167
14.9.8 Delegate equality operators.....	167
14.9.9 The is operator	168
14.9.10 The as operator	168
14.10 Logical operators	169
14.10.1 Integer logical operators	169
14.10.2 Enumeration logical operators	169
14.10.3 Boolean logical operators	170
14.11 Conditional logical operators.....	170
14.11.1 Boolean conditional logical operators.....	170
14.11.2 User-defined conditional logical operators	171
14.12 Conditional operator.....	171
14.13 Assignment operators	172



14.13.1 Simple assignment	172
14.13.2 Compound assignment	174
14.13.3 Event assignment	175
14.14 Expression	175
14.15 Constant expressions	175
14.16 Boolean expressions	176
15. Statements	177
15.1 End points and reachability	177
15.2 Blocks	179
15.2.1 Statement lists	179
15.3 The empty statement	179
15.4 Labeled statements	180
15.5 Declaration statements	180
15.5.1 Local variable declarations	180
15.5.2 Local constant declarations	181
15.6 Expression statements	182
15.7 Selection statements	182
15.7.1 The <code>if</code> statement	182
15.7.2 The <code>switch</code> statement	183
15.8 Iteration statements	186
15.8.1 The <code>while</code> statement	186
15.8.2 The <code>do</code> statement	187
15.8.3 The <code>for</code> statement	187
15.8.4 The <code>foreach</code> statement	188
15.9 Jump statements	190
15.9.1 The <code>break</code> statement	191
15.9.2 The <code>continue</code> statement	192
15.9.3 The <code>goto</code> statement	192
15.9.4 The <code>return</code> statement	193
15.9.5 The <code>throw</code> statement	194
15.10 The <code>try</code> statement	195
15.11 The <code>checked</code> and <code>unchecked</code> statements	197
15.12 The <code>lock</code> statement	198
15.13 The <code>using</code> statement	198
16. Namespaces	201
16.1 Compilation units	201
16.2 Namespace declarations	201
16.3 Using directives	202
16.3.1 Using alias directives	203
16.3.2 Using namespace directives	205
16.4 Namespace members	207
16.5 Type declarations	207
17. Classes	209
17.1 Class declarations	209
17.1.1 Class modifiers	209
17.1.2 Class base specification	210
17.1.3 Class body	211
17.2 Class members	212
17.2.1 Inheritance	213
17.2.2 The <code>new</code> modifier	213
17.2.3 Access modifiers	213
17.2.4 Constituent types	214
17.2.5 Static and instance members	214

17.2.6 Nested types	215
17.2.7 Reserved member names	217
17.3 Constants	219
17.4 Fields	220
17.4.1 Static and instance fields.....	221
17.4.2 Readonly fields	222
17.4.3 Volatile fields.....	223
17.4.4 Field initialization	224
17.4.5 Variable initializers.....	224
17.5 Methods	227
17.5.1 Method parameters	228
17.5.2 Static and instance methods.....	233
17.5.3 Virtual methods.....	233
17.5.4 Override methods.....	235
17.5.5 Sealed methods	236
17.5.6 Abstract methods	237
17.5.7 External methods	238
17.5.8 Method body	239
17.5.9 Method overloading.....	239
17.6 Properties.....	239
17.6.1 Static and instance properties.....	240
17.6.2 Accessors	240
17.6.3 Virtual, sealed, override, and abstract accessors.....	245
17.7 Events	246
17.7.1 Field-like events.....	248
17.7.2 Event accessors.....	249
17.7.3 Static and instance events.....	250
17.7.4 Virtual, sealed, override, and abstract accessors.....	250
17.8 Indexers	251
17.8.1 Indexer overloading.....	254
17.9 Operators	254
17.9.1 Unary operators.....	255
17.9.2 Binary operators.....	256
17.9.3 Conversion operators	256
17.10 Instance constructors	258
17.10.1 Constructor initializers.....	259
17.10.2 Instance variable initializers	259
17.10.3 Constructor execution	259
17.10.4 Default constructors.....	261
17.10.5 Private constructors.....	262
17.10.6 Optional instance constructor parameters.....	262
17.11 Static constructors	262
17.12 Destructors.....	264
18. Structs	267
18.1 Struct declarations	267
18.1.1 Struct modifiers.....	267
18.1.2 Struct interfaces	267
18.1.3 Struct body.....	268
18.2 Struct members.....	268
18.3 Class and struct differences.....	268
18.3.1 Value semantics	268
18.3.2 Inheritance	269
18.3.3 Assignment	269
18.3.4 Default values	269

iTech STANDARD PREVIEW
(standards.iteh.ai)

18.3.5 Boxing and unboxing	270
18.3.6 Meaning of <code>this</code>	270
18.3.7 Field initializers	270
18.3.8 Constructors	270
18.3.9 Destructors	271
18.3.10 Static constructors	271
18.4 Struct examples	271
18.4.1 Database integer type	272
18.4.2 Database boolean type	273
19. Arrays.....	275
19.1 Array types	275
19.1.1 The <code>System.Array</code> type.....	276
19.2 Array creation	276
19.3 Array element access	276
19.4 Array members	276
19.5 Array covariance	276
19.6 Array initializers	277
20. Interfaces.....	279
20.1 Interface declarations	279
20.1.1 Interface modifiers	279
20.1.2 Base interfaces	279
20.1.3 Interface body	280
20.2 Interface members	280
20.2.1 Interface methods	281
20.2.2 Interface properties	281
20.2.3 Interface events	282
20.2.4 Interface indexers	282
20.2.5 Interface member access	282
20.3 Fully qualified interface member names	283
20.4 Interface implementations	284
20.4.1 Explicit interface member implementations	285
20.4.2 Interface mapping	286
20.4.3 Interface implementation inheritance.....	289
20.4.4 Interface re-implementation.....	290
20.4.5 Abstract classes and interfaces.....	291
21. Enums.....	293
21.1 Enum declarations	293
21.2 Enum modifiers	293
21.3 Enum members	294
21.4 Enum values and operations	296
22. Delegates.....	297
22.1 Delegate declarations.....	297
22.2 Delegate instantiation	299
22.3 Delegate invocation	299
23. Exceptions	303
23.1 Causes of exceptions	303
23.2 The <code>System.Exception</code> class.....	303
23.3 How exceptions are handled.....	303
23.4 Common Exception Classes	304
24. Attributes	305

24.1 Attribute classes.....	305
24.1.1 Attribute usage.....	305
24.1.2 Positional and named parameters.....	306
24.1.3 Attribute parameter types.....	307
24.2 Attribute specification.....	307
24.3 Attribute instances.....	311
24.3.1 Compilation of an attribute.....	311
24.3.2 Run-time retrieval of an attribute instance.....	311
24.4 Reserved attributes.....	312
24.4.1 The <code>AttributeUsage</code> attribute.....	312
24.4.2 The <code>Conditional</code> attribute.....	312
24.4.3 The <code>Obsolete</code> attribute.....	314
25. Unsafe code.....	317
25.1 Unsafe contexts.....	317
25.2 Pointer types.....	319
25.3 Fixed and moveable variables.....	322
25.4 Pointer conversions.....	322
25.5 Pointers in expressions.....	323
25.5.1 Pointer indirection.....	324
25.5.2 Pointer member access.....	324
25.5.3 Pointer element access.....	325
25.5.4 The address-of operator.....	325
25.5.5 Pointer increment and decrement.....	326
25.5.6 Pointer arithmetic.....	326
25.5.7 Pointer comparison.....	327
25.5.8 The <code>sizeof</code> operator.....	327
25.6 The <code>fixed</code> statement.....	328
25.7 Stack allocation.....	331
25.8 Dynamic memory allocation.....	332
A. Grammar.....	335
A.1 Lexical grammar.....	335
A.1.1 Line terminators.....	335
A.1.2 White space.....	335
A.1.3 Comments.....	335
A.1.4 Tokens.....	336
A.1.5 Unicode character escape sequences.....	336
A.1.6 Identifiers.....	336
A.1.7 Keywords.....	337
A.1.8 Literals.....	338
A.1.9 Operators and punctuators.....	339
A.1.10 Pre-processing directives.....	340
A.2 Syntactic grammar.....	341
A.2.1 Basic concepts.....	341
A.2.2 Types.....	342
A.2.3 Variables.....	343
A.2.4 Expressions.....	343
A.2.5 Statements.....	346
A.2.6 Classes.....	350
A.2.7 Structs.....	355
A.2.8 Arrays.....	355
A.2.9 Interfaces.....	356
A.2.10 Enums.....	357
A.2.11 Delegates.....	357

A.2.12 Attributes.....	358
A.3 Grammar extensions for unsafe code	359
B. Portability issues	361
B.1 Undefined behavior.....	361
B.2 Implementation-defined behavior.....	361
B.3 Unspecified behavior	362
B.4 Other Issues.....	362
C. Naming guidelines	363
C.1 Capitalization styles.....	363
C.1.1 Pascal casing	363
C.1.2 Camel casing	363
C.1.3 All uppercase.....	363
C.1.4 Capitalization summary.....	363
C.2 Word choice	364
C.3 Namespaces	364
C.4 Classes	364
C.5 Interfaces.....	365
C.6 Enums	365
C.7 Static fields	366
C.8 Parameters.....	366
C.9 Methods	366
C.10 Properties	366
C.11 Events.....	367
C.12 Case sensitivity	367
C.13 Avoiding type name confusion.....	368
D. Standard Library	369
E. Documentation Comments	433
E.1 Introduction.....	433
E.2 Recommended tags	434
E.2.1 <c>	434
E.2.2 <code>	435
E.2.3 <example>.....	435
E.2.4 <exception>.....	435
E.2.5 <list>.....	436
E.2.6 <para>.....	437
E.2.7 <param>	437
E.2.8 <paramref>	437
E.2.9 <permission>	438
E.2.10 <remarks>.....	438
E.2.11 <returns>	438
E.2.12 <see>	439
E.2.13 <seealso>.....	439
E.2.14 <summary>.....	440
E.2.15 <value>.....	440
E.3 Processing the documentation file	440
E.3.1 ID string format	440
E.3.2 ID string examples.....	441
E.4 An example	444
E.4.1 C# source code	444
E.4.2 Resulting XML.....	446
F. Index	449

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.

(standards.iteh.ai)

[ISO/IEC 23270:2003](https://standards.iteh.ai/catalog/standards/sist/b96b2fb-28fe-4204-89dd-7eab24f88245/iso-iec-23270-2003)

<https://standards.iteh.ai/catalog/standards/sist/b96b2fb-28fe-4204-89dd-7eab24f88245/iso-iec-23270-2003>

Introduction

This International Standard is based on a submission from Hewlett-Packard, Intel, and Microsoft, that describes 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.

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 regards to memory and processing power requirements, the language was not intended to compete directly on performance and size with C or assembly language.

The development of this standard started in November 2000.

It is expected there will be future revisions to this standard, primarily to add new functionality.

iTeh STANDARD PREVIEW
(standards.iteh.ai)

ISO/IEC 23270:2003

<https://standards.iteh.ai/catalog/standards/sist/b96f32fb-28fe-4204-89dd-7eab24f88245/iso-iec-23270-2003>

Information technology — C# Language Specification

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.