
**Information technology — Programming
languages — Ada**

Technologies de l'information — Langages de programmation — Ada

**iTeh STANDARD PREVIEW
(standards.iteh.ai)**

[ISO/IEC 8652:2012](https://standards.iteh.ai/catalog/standards/sist/e0057afb-672c-4418-8b15-61c2cd98853e/iso-iec-8652-2012)

<https://standards.iteh.ai/catalog/standards/sist/e0057afb-672c-4418-8b15-61c2cd98853e/iso-iec-8652-2012>

iTeh STANDARD PREVIEW (standards.iteh.ai)

ISO/IEC 8652:2012

<https://standards.iteh.ai/catalog/standards/sist/e0057afb-672c-4418-8b15-61c2cd98853e/iso-iec-8652-2012>



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2012

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

Table of Contents.....	i
Foreword	xi
Introduction.....	xii
1 General	1
1.1 Scope	1
1.1.1 Extent.....	1
1.1.2 Structure.....	2
1.1.3 Conformity of an Implementation with the Standard	4
1.1.4 Method of Description and Syntax Notation	5
1.1.5 Classification of Errors	6
1.2 Normative References	7
1.3 Terms and Definitions	8
2 Lexical Elements.....	9
2.1 Character Set	9
2.2 Lexical Elements, Separators, and Delimiters	11
2.3 Identifiers.....	12
2.4 Numeric Literals.....	13
2.4.1 Decimal Literals.....	13
2.4.2 Based Literals.....	13
2.5 Character Literals	14
2.6 String Literals.....	14
2.7 Comments	15
2.8 Pragmas	15
2.9 Reserved Words	17
3 Declarations and Types.....	19
3.1 Declarations	19
3.2 Types and Subtypes.....	20
3.2.1 Type Declarations.....	21
3.2.2 Subtype Declarations	23
3.2.3 Classification of Operations	24
3.2.4 Subtype Predicates	24
3.3 Objects and Named Numbers.....	26
3.3.1 Object Declarations.....	28
3.3.2 Number Declarations	31
3.4 Derived Types and Classes	31
3.4.1 Derivation Classes	34
3.5 Scalar Types.....	35
3.5.1 Enumeration Types	40
3.5.2 Character Types	41
3.5.3 Boolean Types	42
3.5.4 Integer Types	42
3.5.5 Operations of Discrete Types.....	44
3.5.6 Real Types.....	45
3.5.7 Floating Point Types	46
3.5.8 Operations of Floating Point Types	48
3.5.9 Fixed Point Types.....	48
3.5.10 Operations of Fixed Point Types	50

3.6 Array Types	51
3.6.1 Index Constraints and Discrete Ranges	54
3.6.2 Operations of Array Types	55
3.6.3 String Types	56
3.7 Discriminants	56
3.7.1 Discriminant Constraints	59
3.7.2 Operations of Discriminated Types	60
3.8 Record Types	60
3.8.1 Variant Parts and Discrete Choices	62
3.9 Tagged Types and Type Extensions	64
3.9.1 Type Extensions	67
3.9.2 Dispatching Operations of Tagged Types	68
3.9.3 Abstract Types and Subprograms	71
3.9.4 Interface Types	72
3.10 Access Types	75
3.10.1 Incomplete Type Declarations	77
3.10.2 Operations of Access Types	79
3.11 Declarative Parts	85
3.11.1 Completions of Declarations	85
4 Names and Expressions	87
4.1 Names	87
4.1.1 Indexed Components	88
4.1.2 Slices	89
4.1.3 Selected Components	89
4.1.4 Attributes	91
4.1.5 User-Defined References	92
4.1.6 User-Defined Indexing	93
4.2 Literals	95
4.3 Aggregates	96
4.3.1 Record Aggregates	96
4.3.2 Extension Aggregates	98
4.3.3 Array Aggregates	99
4.4 Expressions	102
4.5 Operators and Expression Evaluation	103
4.5.1 Logical Operators and Short-circuit Control Forms	104
4.5.2 Relational Operators and Membership Tests	105
4.5.3 Binary Adding Operators	109
4.5.4 Unary Adding Operators	110
4.5.5 Multiplying Operators	110
4.5.6 Highest Precedence Operators	112
4.5.7 Conditional Expressions	113
4.5.8 Quantified Expressions	114
4.6 Type Conversions	115
4.7 Qualified Expressions	119
4.8 Allocators	120
4.9 Static Expressions and Static Subtypes	122
4.9.1 Statically Matching Constraints and Subtypes	125
5 Statements	127
5.1 Simple and Compound Statements - Sequences of Statements	127
5.2 Assignment Statements	128
5.3 If Statements	129
5.4 Case Statements	130

5.5 Loop Statements.....	131
5.5.1 User-Defined Iterator Types	133
5.5.2 Generalized Loop Iteration	134
5.6 Block Statements.....	136
5.7 Exit Statements.....	137
5.8 Goto Statements	137
6 Subprograms.....	139
6.1 Subprogram Declarations.....	139
6.1.1 Preconditions and Postconditions	142
6.2 Formal Parameter Modes	144
6.3 Subprogram Bodies	145
6.3.1 Conformance Rules.....	146
6.3.2 Inline Expansion of Subprograms	148
6.4 Subprogram Calls.....	148
6.4.1 Parameter Associations.....	150
6.5 Return Statements.....	153
6.5.1 Nonreturning Procedures	155
6.6 Overloading of Operators	156
6.7 Null Procedures	157
6.8 Expression Functions	158
7 Packages	159
7.1 Package Specifications and Declarations.....	159
7.2 Package Bodies.....	160
7.3 Private Types and Private Extensions.....	161
7.3.1 Private Operations.....	163
7.3.2 Type Invariants	165
7.4 Deferred Constants	167
7.5 Limited Types.....	168
7.6 Assignment and Finalization.....	170
7.6.1 Completion and Finalization.....	172
8 Visibility Rules	175
8.1 Declarative Region	175
8.2 Scope of Declarations	176
8.3 Visibility	177
8.3.1 Overriding Indicators	179
8.4 Use Clauses	180
8.5 Renaming Declarations.....	181
8.5.1 Object Renaming Declarations	182
8.5.2 Exception Renaming Declarations	183
8.5.3 Package Renaming Declarations	183
8.5.4 Subprogram Renaming Declarations	184
8.5.5 Generic Renaming Declarations	186
8.6 The Context of Overload Resolution.....	186
9 Tasks and Synchronization.....	189
9.1 Task Units and Task Objects.....	189
9.2 Task Execution - Task Activation.....	192
9.3 Task Dependence - Termination of Tasks	193
9.4 Protected Units and Protected Objects	194
9.5 Intertask Communication.....	197
9.5.1 Protected Subprograms and Protected Actions.....	199
9.5.2 Entries and Accept Statements.....	200

9.5.3 Entry Calls.....	203
9.5.4 Requeue Statements.....	205
9.6 Delay Statements, Duration, and Time.....	207
9.6.1 Formatting, Time Zones, and other operations for Time.....	209
9.7 Select Statements.....	215
9.7.1 Selective Accept.....	215
9.7.2 Timed Entry Calls.....	217
9.7.3 Conditional Entry Calls.....	218
9.7.4 Asynchronous Transfer of Control.....	219
9.8 Abort of a Task - Abort of a Sequence of Statements.....	220
9.9 Task and Entry Attributes.....	221
9.10 Shared Variables.....	222
9.11 Example of Tasking and Synchronization.....	223
10 Program Structure and Compilation Issues.....	225
10.1 Separate Compilation.....	225
10.1.1 Compilation Units - Library Units.....	225
10.1.2 Context Clauses - With Clauses.....	228
10.1.3 Subunits of Compilation Units.....	230
10.1.4 The Compilation Process.....	232
10.1.5 Pragmas and Program Units.....	233
10.1.6 Environment-Level Visibility Rules.....	234
10.2 Program Execution.....	234
10.2.1 Elaboration Control.....	236
11 Exceptions.....	241
11.1 Exception Declarations.....	241
11.2 Exception Handlers.....	242
11.3 Raise Statements.....	243
11.4 Exception Handling.....	243
11.4.1 The Package Exceptions.....	244
11.4.2 Pragmas Assert and Assertion_Policy.....	246
11.4.3 Example of Exception Handling.....	248
11.5 Suppressing Checks.....	249
11.6 Exceptions and Optimization.....	252
12 Generic Units.....	253
12.1 Generic Declarations.....	253
12.2 Generic Bodies.....	255
12.3 Generic Instantiation.....	256
12.4 Formal Objects.....	258
12.5 Formal Types.....	259
12.5.1 Formal Private and Derived Types.....	261
12.5.2 Formal Scalar Types.....	263
12.5.3 Formal Array Types.....	263
12.5.4 Formal Access Types.....	264
12.5.5 Formal Interface Types.....	265
12.6 Formal Subprograms.....	265
12.7 Formal Packages.....	267
12.8 Example of a Generic Package.....	269
13 Representation Issues.....	273
13.1 Operational and Representation Aspects.....	273
13.1.1 Aspect Specifications.....	276
13.2 Packed Types.....	278

13.3 Operational and Representation Attributes	279
13.4 Enumeration Representation Clauses	285
13.5 Record Layout.....	286
13.5.1 Record Representation Clauses	286
13.5.2 Storage Place Attributes	288
13.5.3 Bit Ordering.....	289
13.6 Change of Representation	290
13.7 The Package System	291
13.7.1 The Package System.Storage_Elements	293
13.7.2 The Package System.Address_To_Access_Conversions.....	294
13.8 Machine Code Insertions	294
13.9 Unchecked Type Conversions.....	295
13.9.1 Data Validity	296
13.9.2 The Valid Attribute.....	297
13.10 Unchecked Access Value Creation	298
13.11 Storage Management	298
13.11.1 Storage Allocation Attributes.....	301
13.11.2 Unchecked Storage Deallocation.....	302
13.11.3 Default Storage Pools	303
13.11.4 Storage Subpools.....	304
13.11.5 Subpool Reclamation.....	306
13.11.6 Storage Subpool Example	307
13.12 Pragma Restrictions and Pragma Profile	309
13.12.1 Language-Defined Restrictions and Profiles.....	310
13.13 Streams.....	312
13.13.1 The Package Streams.....	312
13.13.2 Stream-Oriented Attributes	313
13.14 Freezing Rules	318
The Standard Libraries.....	321
Annex A (normative) Predefined Language Environment.....	323
A.1 The Package Standard.....	326
A.2 The Package Ada	330
A.3 Character Handling	330
A.3.1 The Packages Characters, Wide_Characters, and Wide_Wide_Characters	330
A.3.2 The Package Characters.Handling.....	331
A.3.3 The Package Characters.Latin_1.....	333
A.3.4 The Package Characters.Conversions	338
A.3.5 The Package Wide_Characters.Handling	340
A.3.6 The Package Wide_Wide_Characters.Handling.....	342
A.4 String Handling	343
A.4.1 The Package Strings.....	343
A.4.2 The Package Strings.Maps	343
A.4.3 Fixed-Length String Handling.....	346
A.4.4 Bounded-Length String Handling	354
A.4.5 Unbounded-Length String Handling	361
A.4.6 String-Handling Sets and Mappings	366
A.4.7 Wide_String Handling.....	366
A.4.8 Wide_Wide_String Handling	368
A.4.9 String Hashing	371
A.4.10 String Comparison.....	372
A.4.11 String Encoding	373
A.5 The Numerics Packages.....	378

A.5.1 Elementary Functions	378
A.5.2 Random Number Generation	381
A.5.3 Attributes of Floating Point Types	386
A.5.4 Attributes of Fixed Point Types	390
A.6 Input-Output	390
A.7 External Files and File Objects	390
A.8 Sequential and Direct Files	391
A.8.1 The Generic Package Sequential_IO	392
A.8.2 File Management.....	393
A.8.3 Sequential Input-Output Operations	395
A.8.4 The Generic Package Direct_IO	395
A.8.5 Direct Input-Output Operations	396
A.9 The Generic Package Storage_IO	397
A.10 Text Input-Output.....	397
A.10.1 The Package Text_IO.....	399
A.10.2 Text File Management	403
A.10.3 Default Input, Output, and Error Files.....	404
A.10.4 Specification of Line and Page Lengths.....	405
A.10.5 Operations on Columns, Lines, and Pages.....	406
A.10.6 Get and Put Procedures.....	409
A.10.7 Input-Output of Characters and Strings	410
A.10.8 Input-Output for Integer Types	412
A.10.9 Input-Output for Real Types	414
A.10.10 Input-Output for Enumeration Types.....	416
A.10.11 Input-Output for Bounded Strings	417
A.10.12 Input-Output for Unbounded Strings.....	418
A.11 Wide Text Input-Output and Wide Wide Text Input-Output.....	419
A.12 Stream Input-Output	420
A.12.1 The Package Streams.Stream_IO.....	420
A.12.2 The Package Text_IO.Text_Streams	422
A.12.3 The Package Wide_Text_IO.Text_Streams	423
A.12.4 The Package Wide_Wide_Text_IO.Text_Streams.....	423
A.13 Exceptions in Input-Output.....	423
A.14 File Sharing	425
A.15 The Package Command_Line	425
A.16 The Package Directories	426
A.16.1 The Package Directories.Hierarchical_File_Names	433
A.17 The Package Environment_Variables	435
A.18 Containers	438
A.18.1 The Package Containers	438
A.18.2 The Generic Package Containers.Vectors	438
A.18.3 The Generic Package Containers.Doubly_Linked_Lists.....	454
A.18.4 Maps.....	465
A.18.5 The Generic Package Containers.Hashed_Maps	471
A.18.6 The Generic Package Containers.Ordered_Maps	475
A.18.7 Sets	479
A.18.8 The Generic Package Containers.Hashed_Sets	486
A.18.9 The Generic Package Containers.Ordered_Sets	491
A.18.10 The Generic Package Containers.Multiway_Trees	496
A.18.11 The Generic Package Containers.Indefinite_Vectors	510
A.18.12 The Generic Package Containers.Indefinite_Doubly_Linked_Lists.....	510
A.18.13 The Generic Package Containers.Indefinite_Hashed_Maps	511
A.18.14 The Generic Package Containers.Indefinite_Ordered_Maps.....	511
A.18.15 The Generic Package Containers.Indefinite_Hashed_Sets	511

A.18.16	The Generic Package Containers.Indefinite_Ordered_Sets	512
A.18.17	The Generic Package Containers.Indefinite_Multiway_Trees	512
A.18.18	The Generic Package Containers.Indefinite_Holders.....	512
A.18.19	The Generic Package Containers.Bounded_Vectors	516
A.18.20	The Generic Package Containers.Bounded_Doubly_Linked_Lists	516
A.18.21	The Generic Package Containers.Bounded_Hashed_Maps	518
A.18.22	The Generic Package Containers.Bounded_Ordered_Maps	519
A.18.23	The Generic Package Containers.Bounded_Hashed_Sets.....	520
A.18.24	The Generic Package Containers.Bounded_Ordered_Sets.....	521
A.18.25	The Generic Package Containers.Bounded_Multiway_Trees.....	522
A.18.26	Array Sorting	524
A.18.27	The Generic Package Containers.Synchronized_Queue_Interfaces	525
A.18.28	The Generic Package Containers.Unbounded_Synchronized_Queues ..	526
A.18.29	The Generic Package Containers.Bounded_Synchronized_Queues.....	527
A.18.30	The Generic Package Containers.Unbounded_Priority_Queues	527
A.18.31	The Generic Package Containers.Bounded_Priority_Queues.....	529
A.18.32	Example of Container Use	530
A.19	The Package Locales	532
Annex B (normative)	Interface to Other Languages.....	533
B.1	Interfacing Aspects	533
B.2	The Package Interfaces	536
B.3	Interfacing with C and C++	537
B.3.1	The Package Interfaces.C.Strings	543
B.3.2	The Generic Package Interfaces.C.Pointers	546
B.3.3	Unchecked Union Types	548
B.4	Interfacing with COBOL	550
B.5	Interfacing with Fortran	556
Annex C (normative)	Systems Programming	559
C.1	Access to Machine Operations	559
C.2	Required Representation Support.....	560
C.3	Interrupt Support.....	560
C.3.1	Protected Procedure Handlers	562
C.3.2	The Package Interrupts	564
C.4	Preelaboration Requirements	566
C.5	Pragma Discard_Names	566
C.6	Shared Variable Control	567
C.7	Task Information	569
C.7.1	The Package Task_Identification	569
C.7.2	The Package Task_Attributes	571
C.7.3	The Package Task_Termination	573
Annex D (normative)	Real-Time Systems	575
D.1	Task Priorities	575
D.2	Priority Scheduling	577
D.2.1	The Task Dispatching Model	577
D.2.2	Task Dispatching Pragmas	578
D.2.3	Preemptive Dispatching	580
D.2.4	Non-Preemptive Dispatching	580
D.2.5	Round Robin Dispatching	582
D.2.6	Earliest Deadline First Dispatching.....	583
D.3	Priority Ceiling Locking	585
D.4	Entry Queuing Policies	587
D.5	Dynamic Priorities.....	588

D.5.1 Dynamic Priorities for Tasks	588
D.5.2 Dynamic Priorities for Protected Objects	589
D.6 Preemptive Abort	590
D.7 Tasking Restrictions	591
D.8 Monotonic Time	593
D.9 Delay Accuracy	596
D.10 Synchronous Task Control	597
D.10.1 Synchronous Barriers	598
D.11 Asynchronous Task Control	599
D.12 Other Optimizations and Determinism Rules	600
D.13 The Ravenscar Profile	601
D.14 Execution Time	602
D.14.1 Execution Time Timers	604
D.14.2 Group Execution Time Budgets	606
D.14.3 Execution Time of Interrupt Handlers	608
D.15 Timing Events	608
D.16 Multiprocessor Implementation	610
D.16.1 Multiprocessor Dispatching Domains	611
Annex E (normative) Distributed Systems	615
E.1 Partitions	615
E.2 Categorization of Library Units	617
E.2.1 Shared Passive Library Units	617
E.2.2 Remote Types Library Units	618
E.2.3 Remote Call Interface Library Units	619
E.3 Consistency of a Distributed System	620
E.4 Remote Subprogram Calls	621
E.4.1 Asynchronous Remote Calls	623
E.4.2 Example of Use of a Remote Access-to-Class-Wide Type	623
E.5 Partition Communication Subsystem	625
Annex F (normative) Information Systems	629
F.1 Machine_Radix Attribute Definition Clause	629
F.2 The Package Decimal	629
F.3 Edited Output for Decimal Types	630
F.3.1 Picture String Formation	632
F.3.2 Edited Output Generation	635
F.3.3 The Package Text_IO.Editing	638
F.3.4 The Package Wide_Text_IO.Editing	641
F.3.5 The Package Wide_Wide_Text_IO.Editing	642
Annex G (normative) Numerics	643
G.1 Complex Arithmetic	643
G.1.1 Complex Types	643
G.1.2 Complex Elementary Functions	647
G.1.3 Complex Input-Output	651
G.1.4 The Package Wide_Text_IO.Complex_IO	653
G.1.5 The Package Wide_Wide_Text_IO.Complex_IO	653
G.2 Numeric Performance Requirements	653
G.2.1 Model of Floating Point Arithmetic	654
G.2.2 Model-Oriented Attributes of Floating Point Types	655
G.2.3 Model of Fixed Point Arithmetic	656
G.2.4 Accuracy Requirements for the Elementary Functions	658
G.2.5 Performance Requirements for Random Number Generation	659
G.2.6 Accuracy Requirements for Complex Arithmetic	660

G.3 Vector and Matrix Manipulation.....	662
G.3.1 Real Vectors and Matrices	662
G.3.2 Complex Vectors and Matrices	667
Annex H (normative) High Integrity Systems.....	677
H.1 Pragma Normalize_Scalars	677
H.2 Documentation of Implementation Decisions	678
H.3 Reviewable Object Code	678
H.3.1 Pragma Reviewable	678
H.3.2 Pragma Inspection_Point.....	679
H.4 High Integrity Restrictions	680
H.5 Pragma Detect_Blocking.....	682
H.6 Pragma Partition_Elaboration_Policy	682
Annex J (normative) Obsolescent Features	685
J.1 Renamings of Library Units	685
J.2 Allowed Replacements of Characters	685
J.3 Reduced Accuracy Subtypes	686
J.4 The Constrained Attribute.....	686
J.5 ASCII	687
J.6 Numeric_Error.....	687
J.7 At Clauses	687
J.7.1 Interrupt Entries.....	688
J.8 Mod Clauses.....	689
J.9 The Storage_Size Attribute.....	689
J.10 Specific Suppression of Checks	689
J.11 The Class Attribute of Untagged Incomplete Types.....	690
J.12 Pragma Interface.....	690
J.13 Dependence Restriction Identifiers.....	690
J.14 Character and Wide_Character Conversion Functions.....	691
J.15 Aspect-related Pragmas.....	691
J.15.1 Pragma Inline	691
J.15.2 Pragma No_Return	692
J.15.3 Pragma Pack.....	692
J.15.4 Pragma Storage_Size.....	692
J.15.5 Interfacing Pragmas	692
J.15.6 Pragma Unchecked_Union	693
J.15.7 Pragmas Interrupt_Handler and Attach_Handler	694
J.15.8 Shared Variable Pragmas	694
J.15.9 Pragma CPU.....	695
J.15.10 Pragma Dispatching_Domain.....	695
J.15.11 Pragmas Priority and Interrupt_Priority	696
J.15.12 Pragma Relative_Deadline.....	696
J.15.13 Pragma Asynchronous	697
Annex K (informative) Language-Defined Aspects and Attributes	699
K.1 Language-Defined Aspects.....	699
K.2 Language-Defined Attributes	702
Annex L (informative) Language-Defined Pragmas.....	717
Annex M (informative) Summary of Documentation Requirements.....	719
M.1 Specific Documentation Requirements	719
M.2 Implementation-Defined Characteristics	721
M.3 Implementation Advice	726

Annex N (informative) Glossary	735
Annex P (informative) Syntax Summary	741
Annex Q (informative) Language-Defined Entities	769
Q.1 Language-Defined Packages	769
Q.2 Language-Defined Types and Subtypes	771
Q.3 Language-Defined Subprograms	776
Q.4 Language-Defined Exceptions	785
Q.5 Language-Defined Objects	786
Index	791

iTeh STANDARD PREVIEW
(standards.iteh.ai)

[ISO/IEC 8652:2012](https://standards.iteh.ai/catalog/standards/sist/e0057afb-672c-4418-8b15-61c2cd98853e/iso-iec-8652-2012)

<https://standards.iteh.ai/catalog/standards/sist/e0057afb-672c-4418-8b15-61c2cd98853e/iso-iec-8652-2012>

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.

International Standard ISO/IEC 8652 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information Technology* Subcommittee SC22, *Programming languages, their environments and system software interfaces*.

This third edition cancels and replaces the second edition (ISO/IEC 8652:1995), which has been technically revised. It also incorporates the Technical Corrigendum ISO/IEC 8652:1995:COR.1:2001 and Amendment ISO/IEC 8652:1995:AMD 1:2007.

[ISO/IEC 8652:2012](https://standards.iteh.ai/catalog/standards/sist/e0057afb-672c-4418-8b15-61c2cd98853e/iso-iec-8652-2012)

<https://standards.iteh.ai/catalog/standards/sist/e0057afb-672c-4418-8b15-61c2cd98853e/iso-iec-8652-2012>

Introduction

Design Goals

Ada was originally designed with three overriding concerns: program reliability and maintenance, programming as a human activity, and efficiency. The 1995 revision to the language was designed to provide greater flexibility and extensibility, additional control over storage management and synchronization, and standardized packages oriented toward supporting important application areas, while at the same time retaining the original emphasis on reliability, maintainability, and efficiency. This third edition provides further flexibility and adds more standardized packages within the framework provided by the 1995 revision.

The need for languages that promote reliability and simplify maintenance is well established. Hence emphasis was placed on program readability over ease of writing. For example, the rules of the language require that program variables be explicitly declared and that their type be specified. Since the type of a variable is invariant, compilers can ensure that operations on variables are compatible with the properties intended for objects of the type. Furthermore, error-prone notations have been avoided, and the syntax of the language avoids the use of encoded forms in favor of more English-like constructs. Finally, the language offers support for separate compilation of program units in a way that facilitates program development and maintenance, and which provides the same degree of checking between units as within a unit.

Concern for the human programmer was also stressed during the design. Above all, an attempt was made to keep to a relatively small number of underlying concepts integrated in a consistent and systematic way while continuing to avoid the pitfalls of excessive involution. The design especially aims to provide language constructs that correspond intuitively to the normal expectations of users.

Like many other human activities, the development of programs is becoming ever more decentralized and distributed. Consequently, the ability to assemble a program from independently produced software components continues to be a central idea in the design. The concepts of packages, of private types, and of generic units are directly related to this idea, which has ramifications in many other aspects of the language. An allied concern is the maintenance of programs to match changing requirements; type extension and the hierarchical library enable a program to be modified while minimizing disturbance to existing tested and trusted components.

No language can avoid the problem of efficiency. Languages that require over-elaborate compilers, or that lead to the inefficient use of storage or execution time, force these inefficiencies on all machines and on all programs. Every construct of the language was examined in the light of present implementation techniques. Any proposed construct whose implementation was unclear or that required excessive machine resources was rejected.

Language Summary

An Ada program is composed of one or more program units. Program units may be subprograms (which define executable algorithms), packages (which define collections of entities), task units (which define concurrent computations), protected units (which define operations for the coordinated sharing of data between tasks), or generic units (which define parameterized forms of packages and subprograms). Each program unit normally consists of two parts: a specification, containing the information that must be visible to other units, and a body, containing the implementation details, which need not be visible to other units. Most program units can be compiled separately.

This distinction of the specification and body, and the ability to compile units separately, allows a program to be designed, written, and tested as a set of largely independent software components.

An Ada program will normally make use of a library of program units of general utility. The language provides means whereby individual organizations can construct their own libraries. All libraries are structured in a hierarchical manner; this enables the logical decomposition of a subsystem into

individual components. The text of a separately compiled program unit must name the library units it requires.

Program Units

A subprogram is the basic unit for expressing an algorithm. There are two kinds of subprograms: procedures and functions. A procedure is the means of invoking a series of actions. For example, it may read data, update variables, or produce some output. It may have parameters, to provide a controlled means of passing information between the procedure and the point of call. A function is the means of invoking the computation of a value. It is similar to a procedure, but in addition will return a result.

A package is the basic unit for defining a collection of logically related entities. For example, a package can be used to define a set of type declarations and associated operations. Portions of a package can be hidden from the user, thus allowing access only to the logical properties expressed by the package specification.

Subprogram and package units may be compiled separately and arranged in hierarchies of parent and child units giving fine control over visibility of the logical properties and their detailed implementation.

A task unit is the basic unit for defining a task whose sequence of actions may be executed concurrently with those of other tasks. Such tasks may be implemented on multicomputers, multiprocessors, or with interleaved execution on a single processor. A task unit may define either a single executing task or a task type permitting the creation of any number of similar tasks.

A protected unit is the basic unit for defining protected operations for the coordinated use of data shared between tasks. Simple mutual exclusion is provided automatically, and more elaborate sharing protocols can be defined. A protected operation can either be a subprogram or an entry. A protected entry specifies a Boolean expression (an entry barrier) that must be True before the body of the entry is executed. A protected unit may define a single protected object or a protected type permitting the creation of several similar objects.

Declarations and Statements

The body of a program unit generally contains two parts: a declarative part, which defines the logical entities to be used in the program unit, and a sequence of statements, which defines the execution of the program unit.

The declarative part associates names with declared entities. For example, a name may denote a type, a constant, a variable, or an exception. A declarative part also introduces the names and parameters of other nested subprograms, packages, task units, protected units, and generic units to be used in the program unit.

The sequence of statements describes a sequence of actions that are to be performed. The statements are executed in succession (unless a transfer of control causes execution to continue from another place).

An assignment statement changes the value of a variable. A procedure call invokes execution of a procedure after associating any actual parameters provided at the call with the corresponding formal parameters.

Case statements and if statements allow the selection of an enclosed sequence of statements based on the value of an expression or on the value of a condition.

The loop statement provides the basic iterative mechanism in the language. A loop statement specifies that a sequence of statements is to be executed repeatedly as directed by an iteration scheme, or until an exit statement is encountered.

A block statement comprises a sequence of statements preceded by the declaration of local entities used by the statements.