

INTERNATIONAL STANDARD

ISO/IEC
10514-1

First edition
1996-06-01

Information technology — Programming languages —

Part 1: **Modula-2, Base Language**

*Technologies de l'information — Langages de programmation —
Partie 1: Modula-2, langage de base*



Reference number
ISO/IEC 10514-1:1996(E)

Contents

	Page
Foreword	xiv
Introduction	xv
Section 1 : Scope	1
1.1 Goals	1
1.2 Specifications included in this part of ISO/IEC 10514	1
1.3 Specifications not within the scope of this part of ISO/IEC 10514	1
Section 2 : Normative References	3
Section 3 : Definitions, Structure, and Conventions	4
3.1 Definitions	4
3.2 Structure of the Formal Definition	6
3.3 Conventions	8
3.3.1 Informative text	8
3.3.2 Typographical conventions	8
3.3.3 References to pervasive identifiers	9
3.3.4 Interpretation of non-VDM-SL notations	9
3.3.5 References to formal parameter names	9
Section 4 : Requirements for Implementations	10
4.1 Translation	10
4.2 Source Code Representation	10
4.3 Ordering of Declarations	11
4.4 Predefined Entities	11
4.5 Library Modules	11
4.6 Errors	12
4.7 Exceptions	12
4.8 Implementation-dependencies	13
4.9 Documentation	13
4.10 Statement of Compliance	14
4.11 Minimum requirements	14

Section 5 : The Lexis	16
5.1 Source Code Structure	16
5.2 Word tokens	16
5.2.1 Identifiers	16
5.2.2 Portable Identifiers	16
5.2.3 Full Identifiers	16
5.2.4 Pervasive Identifiers	16
5.2.5 Keywords	17
5.3 Symbols and Operators	17
5.3.1 Required Symbols	17
5.3.2 Symbols with alternatives	18
5.3.3 Operators	18
5.4 Constant Literals	20
5.4.1 Whole Number Literals	20
5.4.2 Real Literals	20
5.4.3 String Literals	20
5.5 Separators	21
5.5.1 White Space	21
5.5.2 Comments	21
5.5.3 Source Code Directives	21
5.6 Letters and National Characters	21
Section 6 : The Language	23
6.1 Programs, Program Modules, and Separate Modules	23
6.1.1 Programs and Compilation Modules	23
6.1.2 Program Modules	27
6.1.3 Definition Modules	28
6.1.4 Implementation Modules	30
6.1.4.1 Sourced Implementation Modules	31
6.1.4.2 Abstract Implementation Modules	33
6.1.5 Program Module and Separate Module Consistency	34
6.1.5.1 Definition Module and Implementation Module Consistency	34
6.1.5.2 Module Compilation Order	36
6.1.5.3 Module Dependencies	37
6.1.6 Module Initialization Order	38
6.1.7 Program Termination	40
6.1.8 Import Lists	41
6.1.8.1 Import List	43
6.1.8.2 Simple Imports	43
6.1.8.3 Unqualified Import	44
6.1.8.4 Import Consistency in Definition Modules	45
6.1.8.5 Import Consistency in Module Blocks	46
6.1.9 Export Lists	46
6.1.9.1 Unqualified Exports	46
6.1.9.2 Qualified Exports	47
6.1.10 Implicit Import and Export	48
6.1.11 Protected Modules	50
6.2 Definitions and Declarations	52
6.2.1 Identifiers	52
6.2.2 Qualified Identifiers	52
6.2.3 Definitions	54
6.2.4 Type Definitions	56
6.2.5 Procedure Headings	57
6.2.5.1 Proper Procedure Headings	58
6.2.5.2 Function Procedure Headings	60
6.2.5.3 Formal Parameters	61
6.2.6 Declarations	63

6.2.7	Ordering of Procedure and Module Declarations	66
6.2.8	Constant Declarations	67
6.2.9	Type Declarations	68
6.2.10	Variable Declarations	69
6.2.11	Procedure Declarations	71
6.2.11.1	Proper Procedure Declarations	73
6.2.11.2	Function Procedure Declarations	76
6.2.12	Local Module Declarations	79
6.2.13	Auxiliary Formal Model Definitions	82
6.2.13.1	Identifiers Defined in Definitions	82
6.2.13.2	Identifiers Declared in Declarations	82
6.2.13.3	Identifiers Defined in New Types	83
6.2.13.4	Qualified Identifiers Used in Declarations	84
6.2.13.5	Qualified Identifiers used in Expressions	84
6.2.13.6	Qualified Identifiers Used in Types	87
6.2.13.7	Qualified Identifiers Used in Parameters	88
6.2.13.8	Shielded Environments and Shielded Identifiers	89
6.3	Data Types	91
6.3.1	Type Denoters and Ordinal Type Denoters	91
6.3.2	Type Identifiers	92
6.3.3	New Types	93
6.3.4	Enumeration Types	94
6.3.5	Subrange Types	95
6.3.6	Set Types	97
6.3.7	Packedset Types	98
6.3.8	Pointer Types	99
6.3.9	Procedure Types	100
6.3.9.1	Proper Procedure Types	101
6.3.9.2	Function Procedure Types	101
6.3.9.3	Formal Parameter Type Lists	102
6.3.9.4	Variable Formal Types	103
6.3.9.5	Value Formal Types	104
6.3.10	Formal Types	104
6.3.11	Array Types	106
6.3.12	Record Types	107
6.3.12.1	Fixed Fields	109
6.3.12.2	Variant Fields	109
6.3.13	Storage Allocation	114
6.3.13.1	Allocating Storage for Constructed Types	114
6.3.13.2	Allocating Storage for Elementary Types	115
6.3.13.3	Allocating Storage for Structured (Composite) Types	116
6.3.13.4	Allocating Storage for System Storage Types	120
6.4	Expression and Assignment Compatibility	121
6.4.1	Expression Compatibility	121
6.4.2	Assignment Compatibility	122
6.5	Blocks	126
6.5.1	Proper Procedure Blocks	126
6.5.2	Function Procedure Blocks	129
6.5.3	Module Blocks	131
6.5.4	Static Module Initialization	132
6.5.5	Static Module Finalization	135
6.5.6	Dynamic Module Initialization	136
6.5.7	Dynamic Module Finalization	138
6.5.8	Block Bodies and Exception Handling	140
6.5.8.1	Non-result Bodies	142
6.5.8.2	Result Bodies	144
6.6	Statements	146

6.6.1	Statement Sequences	147
6.6.2	Empty Statements	148
6.6.3	Assignment Statements	148
6.6.4	Procedure Calls	152
6.6.5	Return Statements	153
6.6.5.1	Simple Return Statements	154
6.6.5.2	Function Return Statements	154
6.6.6	Retry Statements	155
6.6.7	With Statements	156
6.6.8	If Statements	157
6.6.9	Case Statements	159
6.6.9.1	Case Alternatives	160
6.6.10	While Statements	161
6.6.11	Repeat Statements	162
6.6.12	Loop Statements	163
6.6.13	Exit Statements	164
6.6.14	For Statements	164
6.6.15	Well-formed Control Variables	168
6.6.15.1	Threatening	169
6.6.16	Auxiliary Functions	176
6.6.16.1	Well-formed Return Statements	176
6.6.16.2	Well-formed Retry Statements	177
6.6.16.3	Well-formed Exit Statements	177
6.6.16.4	Statement Classes of Statement Sequences	179
6.7	Variable Designators	182
6.7.1	Entire Designators	182
6.7.2	Indexed Designators	184
6.7.3	Selected Designators	185
6.7.4	Dereferenced Designators	187
6.8	Expressions	189
6.8.1	Ordinal Expressions	190
6.8.2	Infix Expressions and Operations	190
6.8.2.1	Complex Number Infix Operations	193
6.8.2.2	Real Number Infix Operations	196
6.8.2.3	Whole Number Infix Operations	198
6.8.2.4	Boolean Infix Operations	202
6.8.2.5	Set Infix Operations	203
6.8.2.6	String Literal Infix Operations	205
6.8.2.7	Relational Operations	206
6.8.3	Prefix Expressions	219
6.8.3.1	Arithmetic Prefix Operations	220
6.8.3.2	Boolean Prefix Operations	221
6.8.4	Value Designators	222
6.8.4.1	Entire Values	223
6.8.4.2	Indexed Values	225
6.8.4.3	Selected Values	226
6.8.4.4	Dereferenced Values	227
6.8.5	Function Calls	229
6.8.6	Value Constructors	230
6.8.6.1	Array Constructors	231
6.8.6.2	Record Constructors	236
6.8.6.3	Set Constructors	241
6.8.7	Constant Literals	247
6.8.7.1	Whole Number Literals	248
6.8.7.2	Real Literals	249
6.8.7.3	String Literals	250
6.8.8	Constant Expressions	253

6.8.8.1	Constant Expression Evaluation	257
6.9	Parameter Compatibility and Argument Binding	258
6.9.1	Actual Parameters	258
6.9.1.1	Variable Actual Parameters	259
6.9.1.2	Expression Actual Parameters	260
6.9.1.3	Type Parameters	260
6.9.2	Parameter Matching	261
6.9.3	Parameter Compatibility	261
6.9.3.1	Value Parameter Compatibility	262
6.9.3.2	Variable Parameter Compatibility	263
6.9.3.3	System Storage Parameter Compatibility	264
6.9.4	Argument Binding	265
6.9.4.1	Argument Binding to Value Parameters	266
6.9.4.2	Argument Binding to Variable Parameters	267
6.10	Predefined Types, Standard Procedures and Standard Functions	270
6.10.1	The Pervasive Environment	270
6.10.2	Predefined Types	271
6.10.2.1	Predefined Whole Number Types	271
6.10.2.2	Predefined Real Number Types	272
6.10.2.3	Predefined Complex Number Types	273
6.10.2.4	The Boolean Type	274
6.10.2.5	The Character Type	274
6.10.2.6	Values of the Basic Types	276
6.10.2.7	The String Literal Type	278
6.10.2.8	The Nil Type	279
6.10.2.9	The Bitset Type	279
6.10.2.10	The Proc Type	279
6.10.2.11	The Protection Type	280
6.10.3	Standard Procedures	280
6.10.3.1	Standard Procedure Designators	280
6.10.3.2	The Procedure DEC	281
6.10.3.3	The Procedure DISPOSE	283
6.10.3.4	The Procedure EXCL	284
6.10.3.5	The Procedure HALT	285
6.10.3.6	The Procedure INC	286
6.10.3.7	The Procedure INCL	286
6.10.3.8	The Procedure NEW	287
6.10.4	Standard Functions	288
6.10.4.1	Standard Function Designators	288
6.10.4.2	The Function ABS	290
6.10.4.3	The Function CAP	291
6.10.4.4	The Function CHR	292
6.10.4.5	The Function CMPLX	292
6.10.4.6	The Function FLOAT	293
6.10.4.7	The Function HIGH	294
6.10.4.8	The Function IM	295
6.10.4.9	The Function INT	296
6.10.4.10	The Function LENGTH	297
6.10.4.11	The Function LFLOAT	297
6.10.4.12	The Function MAX	298
6.10.4.13	The Function MIN	299
6.10.4.14	The Function ODD	300
6.10.4.15	The Function ORD	301
6.10.4.16	The Function RE	301
6.10.4.17	The Function SIZE	302
6.10.4.18	The Function TRUNC	303
6.10.4.19	The Function VAL	304

6.10.5	Auxiliary Definitions	306
6.11	The Environment and Auxiliary Formal Definitions	307
6.11.1	Environments	307
6.11.1.1	Constant Values	308
6.11.1.2	Types	308
6.11.1.3	Structures	309
6.11.1.4	Expression Types	311
6.11.1.5	Types for Declared Procedures	311
6.11.1.6	Types for Predefined Procedures	311
6.11.1.7	Module Environments	312
6.11.1.8	Variables	312
6.11.1.9	Protection Domains	313
6.11.1.10	Local Continuations	313
6.11.2	Operations on the Environment	313
6.11.2.1	Functions Accessing the Constants Component of the Environment	313
6.11.2.2	Functions Accessing the Types Component of the Environment	314
6.11.2.3	Functions Accessing the Structures Component of the Environment	315
6.11.2.4	Functions Accessing the Variables Component of the Environment	315
6.11.2.5	Functions Accessing the Procedures Component of the Environment	316
6.11.2.6	Functions Accessing the Modules Component of the Environment	317
6.11.2.7	Functions Accessing the Procedure Level Component of the Environment	317
6.11.2.8	Functions Accessing the Continuations Component of the Environment	318
6.11.2.9	Functions Accessing the Protection Components of the Environment	318
6.11.2.10	Functions Accessing the Environment	319
6.11.2.11	Operations to Update Environments	319
6.11.3	Types	320
6.11.3.1	New Type Names	320
6.11.3.2	Structured (Composite) Types	321
6.11.3.3	Elementary Types	321
6.11.3.4	Scalar Types	322
6.11.3.5	Ordinal Types	322
6.11.3.6	Whole Number Types	323
6.11.3.7	Number Types	324
6.11.3.8	String Types	324
6.11.3.9	System Storage Types	324
6.11.4	Unit Types	325
6.11.4.1	Range Type	325
6.11.4.2	Host Type	325
6.11.4.3	Base Type	325
6.11.4.4	Component Type	325
6.11.4.5	Index Type	326
6.11.4.6	Bound Type	326
6.11.5	Other Type Functions	326
6.11.5.1	Return Types	326
6.11.5.2	Formal Parameter Types	326
6.11.5.3	Auxiliary Functions for Records	326
6.11.5.4	Auxiliary Functions for Arrays	328
6.11.5.5	Auxiliary Functions for Open Arrays	328
6.11.5.6	Auxiliary Functions for Opaque Types	328
6.11.6	Values Associated with a Type	328
6.12	The Storage Model and Auxiliary Formal Definitions	331
6.12.1	The Program State	331
6.12.2	The External Program State	331
6.12.2.1	Operations to Initialize and Terminate the Program State	331
6.12.3	The Store	332
6.12.3.1	Storage	332
6.12.3.2	Values	333

6.12.4	Procedure Denotations	334
6.12.5	Termination	334
6.12.6	Exceptions	334
6.12.7	Interrupt Handlers	335
6.12.8	Operations on Storage	335
6.12.8.1	Operations Accessing Storage	335
6.12.8.2	Storage Aliasing	336
6.12.8.3	Operations to Allocate and Deallocate Storage	337
6.12.8.4	Operations to Allocate and Deallocate Storage Locations	338
6.12.8.5	Access Coroutine Storage	339
6.12.9	Accessing the Procedure Denotation Component of the State	339
6.12.10	Program Initialization and Program Termination	340
6.12.10.1	Program Initialization	340
6.12.10.2	Program Termination	340
6.12.10.3	Program Halting	341
6.12.11	Exception Handling	341
6.12.11.1	Allocation of Exception Sources	342
6.12.12	Interrupt Handlers	343
6.12.13	Accessing the Protection Component of the State	344
6.12.14	Coroutine and Continuations	344
6.12.14.1	The Coroutine Environment	344
6.12.14.2	The Continuations	344
6.12.14.3	Operations on Continuations	345
Section 7	System Modules	347
7.1	The Module SYSTEM	348
7.1.1	The Interface to SYSTEM	348
7.1.2	The Constants of SYSTEM	351
7.1.3	The Types of SYSTEM	352
7.1.3.1	System Storage Types	352
7.1.3.2	The Address Type	353
7.1.4	The Functions of SYSTEM	353
7.1.5	Address Arithmetic Functions	354
7.1.5.1	The Function ADDADR	355
7.1.5.2	The Function SUBADR	356
7.1.5.3	The Function DIFADR	357
7.1.5.4	Properties of the address arithmetic functions	358
7.1.6	Address Construction and Enquiry Functions	359
7.1.6.1	The Function MAKEADR	359
7.1.6.2	The Function ADR	359
7.1.7	Packedset Functions	360
7.1.7.1	Packedset mapping	361
7.1.7.2	The Function ROTATE	361
7.1.7.3	The Function SHIFT	362
7.1.8	The Function CAST	364
7.1.9	The Function TSIZE	365
7.2	The Module COROUTINES	368
7.2.1	The Interface to COROUTINES	368
7.2.2	The Types of COROUTINES	369
7.2.3	The Procedures of COROUTINES	370
7.2.3.1	The Procedure ATTACH	371
7.2.3.2	The Procedure DETACH	371
7.2.3.3	The Procedure IOTRANSFER	372
7.2.3.4	The Procedure LISTEN	373
7.2.3.5	The Procedure NEWCOROUTINE	373
7.2.3.6	The Procedure TRANSFER	376
7.2.4	The Functions of COROUTINES	376

7.2.4.1	The Function HANDLER	377
7.2.4.2	The Function IsATTACHED	378
7.2.4.3	The Function CURRENT	379
7.2.4.4	The Function PROT	379
7.3	The Module EXCEPTIONS	380
7.3.1	The Interface to EXCEPTIONS	380
7.3.2	The Types of EXCEPTIONS	381
7.3.3	The Procedures of EXCEPTIONS	381
7.3.3.1	The Procedure AllocateSource	382
7.3.3.2	The Procedure RAISE	383
7.3.3.3	The Procedure GetMessage	383
7.3.4	The Functions of EXCEPTIONS	384
7.3.4.1	The Function CurrentNumber	385
7.3.4.2	The Function IsCurrentSource	386
7.3.4.3	The Function IsExceptionalExecution	387
7.3.4.4	The Use of EXCEPTIONS	387
7.4	The Module TERMINATION	389
7.4.1	The Interface to TERMINATION	389
7.4.2	The Functions of TERMINATION	389
7.4.2.1	The Function IsTerminating	390
7.4.2.2	The Function HasHalted	391
7.5	The Module M2EXCEPTION	392
7.5.1	The Interface to M2EXCEPTION	392
7.5.2	The Types of M2EXCEPTION	393
7.5.3	The Functions of M2EXCEPTION	393
7.5.3.1	The Function M2Exception	394
7.5.3.2	The Function IsM2Exception	395
7.5.4	Language Exceptions	395
7.5.4.1	Exceptions whose detection is required	396
7.5.4.2	Exceptions whose detection is not required	397
7.5.5	Messages Associated with Language Exceptions	397
7.5.6	Aggregation and Raising of Language Exceptions	398
Section 8	Required Library Modules	399
8.1	The Module Storage	399
8.1.1	The Definition Module of Storage	399
8.1.2	The Procedures of Storage	400
8.1.3	Exceptions Raised by Storage	402
8.1.4	The State for Storage	402
8.1.5	The Initialization of Storage	403
8.1.6	Auxiliary Operations	403
8.2	The Modules LowReal and LowLong	405
8.2.1	The Definition Module of LowReal	405
8.2.2	The Definition Module of LowLong	406
8.2.3	The Constants of LowReal and LowLong	407
8.2.4	The Procedures of LowReal and LowLong	409
8.2.5	Exceptions Raised by LowReal and LowLong	411
8.3	The Module CharClass	412
8.3.1	The Definition Module of CharClass	412
8.3.2	The Procedures of CharClass	412
Section 9	Standard Library Modules	415
9.1	The Mathematical Libraries	415
9.1.1	The Definition Module of RealMath	415
9.1.2	The Definition Module of LongMath	416
9.1.3	The Constants of RealMath and LongMath	417
9.1.4	The Procedures of RealMath and LongMath	417

9.1.5	Exceptions Raised by RealMath and LongMath	420
9.1.6	The State for RealMath and LongMath	421
9.1.7	The Initialization of RealMath and LongMath	421
9.1.8	The Definition Module of ComplexMath	421
9.1.9	The Definition Module of LongComplexMath	422
9.1.10	The Constants of ComplexMath and LongComplexMath	423
9.1.11	The Procedures of ComplexMath and LongComplexMath	423
9.1.12	Exceptions Raised by ComplexMath and LongComplexMath	428
9.1.13	The State for ComplexMath and LongComplexMath	429
9.1.14	The Initialization of ComplexMath and LongComplexMath	429
9.1.15	Auxiliary Functions	429
9.2	The Input/Output Library	430
9.2.1	Standard and Default Channels	431
9.2.1.1	The Module StdChans	431
9.2.2	Reading and Writing of Data using Specified Channels	434
9.2.2.1	The Module TextIO	435
9.2.2.2	The Module WholeIO	439
9.2.2.3	The Modules RealIO and LongIO	442
9.2.2.4	The Module RawIO	447
9.2.2.5	The Module IOConsts	448
9.2.2.6	The Module IOResult	449
9.2.2.7	Auxiliary Formal Model Definitions	450
9.2.3	Reading and Writing of Data using Default Channels	455
9.2.3.1	The Module STextIO	455
9.2.3.2	The Module SWholeIO	456
9.2.3.3	The Modules SRealIO and SLongIO	457
9.2.3.4	The Module SRawIO	459
9.2.3.5	The Module SIOResult	459
9.2.4	The Interface for Device-Independent Channel Operations	461
9.2.4.1	The Module IOChan	461
9.2.5	Obtaining Channels from Device Modules	472
9.2.5.1	The Module ChanConsts	472
9.2.5.2	The Module StreamFile	475
9.2.5.3	The Module SeqFile	483
9.2.5.4	The Module RndFile	497
9.2.5.5	The Module TermFile	511
9.2.5.6	The Module ProgramArgs	522
9.2.5.7	Implementation-defined Functions used in the Device Modules	529
9.2.5.8	Auxiliary Formal Model Definitions	529
9.2.6	The Interface to Channels for New Device Modules	531
9.2.6.1	The Module IOLink	531
9.2.7	The State for the Input/Output Library	536
9.3	Concurrent Programming	544
9.3.1	The Module Processes	544
9.3.1.1	The Definition Module of Processes	546
9.3.1.2	The Procedures of Processes	548
9.3.1.3	Exceptions Raised by Processes	554
9.3.1.4	The State for Processes	555
9.3.1.5	The Initialization of Processes	555
9.3.1.6	Auxiliary Formal Model Definitions	556
9.3.2	The Module Semaphores	559
9.3.2.1	The Definition Module of Semaphores	559
9.3.2.2	The Procedures of Semaphores	559
9.3.2.3	Exceptions Raised by Semaphores	562
9.3.2.4	The State for Semaphores	562
9.3.2.5	The Initialization of Semaphores	563
9.3.2.6	Auxiliary Formal Model Definitions	563

9.4	The Module Strings	564
9.4.1	The Definition Module of Strings	564
9.4.2	The Procedures of Strings	567
9.4.3	Auxiliary Formal Model Definitions	579
9.5	The String Conversions Library	581
9.5.1	Common Data Types	581
9.5.1.1	The Module ConvTypes	581
9.5.2	High-Level String Conversion Modules	582
9.5.2.1	The Module WholeStr	582
9.5.2.2	The Modules RealStr and LongStr	586
9.5.3	Low-Level String Conversion Modules	592
9.5.3.1	The Module WholeConv	592
9.5.3.2	The Modules RealConv and LongConv	598
9.5.4	Auxiliary Definitions	605
9.6	The Module SysClock	607
9.6.1	The Definition Module of SysClock	607
9.6.2	The Types of SysClock	608
9.6.3	The Procedures of SysClock	608
9.6.4	The State for SysClock	609
Section 10:	Auxiliary Formal Model Definitions	611
10.1	Extensions to VDM-SL	611
10.1.1	Replacement Definitions for Combinators	611
10.1.2	The McCarthy Conditional Expression	612
10.2	VDM-SL Functions	614
10.2.1	Set functions	614
10.2.2	Sequence functions	614
10.2.3	Map functions	615
10.2.4	Numerical functions	615
Annexes		
Annex A	Minimum Limit Specifications	616
A.1	The Limit-specification Generators	616
A.1.1	Length of Identifiers	616
A.1.2	Length of String Literals	617
A.1.3	Size of Single Dimension Arrays	617
A.1.4	Number of Dimensions of Multidimension Arrays	618
A.1.5	Depth of Nested Records	619
A.1.6	Number of Sequential Local Modules	620
A.1.7	Depth of Nested Local Modules	620
A.1.8	Number of Enumeration Literals	621
A.1.9	Number of Sequential Procedure Declarations	622
A.1.10	Depth of Nested Procedure Declarations	623
A.1.11	Number of Coroutines	624
A.1.12	Number of Parameters of a Procedure	625
A.1.13	Depth of Nested IF-THEN-ELSEs in THEN-branch	626
A.1.14	Depth of Nested IF-THEN-ELSEs in Both Branches	627
A.1.15	Depth of Nested LOOP Statements	628
A.1.16	Depth of Nested WHILE Statements	629
A.1.17	Depth of Nested REPEAT Statements	630
A.1.18	Depth of Nested FOR Statements	631
A.1.19	Depth of Nested WITH Statements	632
A.1.20	Number of CASE Alternatives with ELSE part	633
A.1.21	Depth of Nested Parentheses in Expressions	634
A.1.22	Depth of Recursion	635
A.1.23	Number of Imported Modules	635

Annex B The Specification of Library Modules	637
B.1 Components of a Library Module Specification	637
B.2 Interpretation of a Library Module Specification	637
Annex C Collected Modula-2 Concrete Syntax	639
C.1 Programs, Program Modules, and Separate Modules	639
C.1.1 Programs and Compilation Modules	639
C.1.2 Program Modules	639
C.1.3 Definition Modules	639
C.1.4 Implementation Modules	639
C.1.4.1 Sourced Implementation Modules	639
C.1.5 Import Lists	639
C.1.5.1 Import List	639
C.1.5.2 Simple Imports	639
C.1.5.3 Unqualified Import	639
C.1.6 Export Lists	640
C.1.6.1 Unqualified Exports	640
C.1.6.2 Qualified Exports	640
C.1.7 Protected Modules	640
C.2 Definitions and Declarations	640
C.2.1 Qualified Identifiers	640
C.2.2 Definitions	640
C.2.3 Type Definitions	640
C.2.4 Procedure Headings	640
C.2.4.1 Proper Procedure Headings	640
C.2.4.2 Function Procedure Headings	641
C.2.4.3 Formal Parameters	641
C.2.5 Declarations	641
C.2.6 Constant Declarations	641
C.2.7 Type Declarations	641
C.2.8 Variable Declarations	641
C.2.9 Procedure Declarations	641
C.2.9.1 Proper Procedure Declarations	642
C.2.9.2 Function Procedure Declarations	642
C.2.10 Local Module Declarations	642
C.3 Data Types	642
C.3.1 Type Denoters and Ordinal Type Denoters	642
C.3.2 Type Identifiers	642
C.3.3 New Types	642
C.3.4 Enumeration Types	642
C.3.5 Subrange Types	642
C.3.6 Set Types	643
C.3.7 Packedset Types	643
C.3.8 Pointer Types	643
C.3.9 Procedure Types	643
C.3.9.1 Proper Procedure Types	643
C.3.9.2 Function Procedure Types	643
C.3.9.3 Formal Parameter Type Lists	643
C.3.9.4 Variable Formal Types	643
C.3.9.5 Value Formal Types	643
C.3.10 Formal Types	643
C.3.11 Array Types	644
C.3.12 Record Types	644
C.3.12.1 Fixed Fields	644
C.3.12.2 Variant Fields	644
C.4 Blocks	644
C.4.1 Proper Procedure Blocks	644

C.4.2	Function Procedure Blocks	644
C.4.3	Module Blocks	645
C.4.4	Block Bodies and Exception Handling	645
C.5	Statements	645
C.5.1	Statement Sequences	645
C.5.2	Empty Statements	645
C.5.3	Assignment Statements	645
C.5.4	Procedure Calls	645
C.5.5	Return Statements	645
C.5.5.1	Simple Return Statements	645
C.5.5.2	Function Return Statements	646
C.5.6	Retry Statements	646
C.5.7	With Statements	646
C.5.8	If Statements	646
C.5.9	Case Statements	646
C.5.9.1	Case Alternatives	646
C.5.10	While Statements	646
C.5.11	Repeat Statements	646
C.5.12	Loop Statements	647
C.5.13	Exit Statements	647
C.5.14	For Statements	647
C.6	Variable Designators	647
C.6.1	Entire Designators	647
C.6.2	Indexed Designators	647
C.6.3	Selected Designators	647
C.6.4	Dereferenced Designators	647
C.7	Expressions	648
C.7.1	Ordinal Expressions	648
C.7.2	Infix Expressions and Operations	648
C.7.3	Value Designators	648
C.7.3.1	Entire Values	648
C.7.3.2	Indexed Values	648
C.7.3.3	Selected Values	648
C.7.3.4	Dereferenced Values	649
C.7.4	Function Calls	649
C.7.5	Value Constructors	649
C.7.5.1	Array Constructors	649
C.7.5.2	Record Constructors	649
C.7.5.3	Set Constructors	649
C.7.6	Constant Literals	650
C.7.7	Constant Expressions	650
C.8	Parameter Compatibility and Argument Binding	650
C.8.1	Actual Parameters	650
C.8.1.1	Type Parameters	650
C.9	Lexical components	650
Annex D	Collected Modula-2 Abstract Syntax	652
Annex E	Modula-2 Glossary	660
Annex F	Participating Individuals and Organisations	674
Index		676

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. 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.

International Standard ISO/IEC 10514-1 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 22, *Programming languages, their environments and system software interfaces*.

ISO/IEC 10514 consists of the following parts, under the general title *Information technology — Programming languages*:

- *Part 1: Modula-2, Base Language*
- *Part 2: Object-oriented Modula-2*

Annexes A and B form an integral part of this part of ISO/IEC 10514. Annexes C to F are for information only.

Introduction

This part of ISO/IEC 10514, Modula-2 (Base language), provides a specification of the form and meaning of Modula-2 programs, and by reference to that specification lays down requirements for Modula-2 implementations.

Modula-2 is a general-purpose programming language, characterized by modules that provide data and procedure encapsulation and hiding. It provides facilities for concurrent programming based on coroutines, and includes facilities for low-level programming. Modula-2 is, therefore, suitable for systems programming. Facilities such as input/output, mathematical operations and string processing are made available through library modules.

Modula-2 is a descendent of Modula whose ancestor is Pascal. All three languages were devised by Professor Niklaus Wirth, ETH, Zürich, Switzerland. The first description of Modula-2 was published in 1980. The language became popular during the early 1980s and many implementations were developed. Because of different interpretations of various descriptions of Modula-2, significant differences in implementations appeared, and in 1984 a group of Modula-2 proponents formed a BSI panel to develop a standard. The BSI panel successfully proposed a new work item to ISO, and the ISO working group met for the first time in 1987. In 1989 the first draft of this standard was published, and in 1992 the second committee draft was published. The base documents for the formulation of this part of ISO/IEC 10514 were the various editions of *Programming in Modula-2* by Niklaus Wirth, published by Springer-Verlag.

This part of ISO/IEC 10514 contains a formal model of Modula-2; for the sake of generality, consistency, and precision, differences have been introduced between the model and the third edition of Professor Wirth's book *Programming in Modula-2* and these are identified either as 'clarifications' or 'changes'. During the standardization of Modula-2 it was recognized that users of the language were demanding extensions to the language devised by Professor Wirth and that implementors were responding by providing them. Several changes and extensions have, therefore, been made in order to prevent further divergence, and to ensure consistency and coherence.

This part of ISO/IEC 10514 uses a variety of notations to specify the requirements contained in it. A standard syntactic metalanguage is used for specifying lexical and syntactic structures. The Vienna Development Method-Specification Language (VDM-SL), which is based upon conventional mathematics, is used to define the semantics of the language in a mathematically rigorous manner. In addition, the semantics of the Modula-2 language, system modules and standard library are also specified using natural language, albeit in a form that is possibly ambiguous. It is intended that the corresponding VDM-SL resolves any such ambiguity. If the natural language requirements and those expressed in the VDM-SL conflict, then there is a mistake in this standard. In many cases, small details of the language semantics may only be resolved by reference to the VDM-SL; by this means the natural language description has been kept relatively simple. A number of typographic and stylistic conventions have been adopted in order to make the natural language description succinct.

The definitions of key terms used in this part of ISO/IEC 10514 are expressed in natural language, with references to VDM-SL where appropriate. The requirements for conforming implementations are given in natural language, with references to VDM-SL where appropriate.

The development of large parts of the formal specification of the semantics of Modula-2 in this part of ISO/IEC 10514 was sponsored by the U.K. Alvey Research Programme. The Alvey sponsored work was carried out at Leicester University Computer Studies Department in collaboration with BSI QA Software Engineering Department.

Information technology — Programming languages —

Part 1:

Modula-2, Base Language

Section 1 : Scope

1.1 Goals

The goals of this part of ISO/IEC 10514 are:

- to provide a rigorous definition of the language Modula-2 and its standard library by providing a mathematical model of both;
- to provide a resolution of differences among interpretations of other descriptions of Modula-2 and its standard library, while endeavouring to preserve investment in existing practice;
- to remove features thought to be redundant, inherently flawed, or inadequate;
- to specify new language and standard library facilities where a need is perceived to exist;
- to maintain the general principles of Modula-2 laid down by its inventor, while allowing for later modernization and standardization.

1.2 Specifications included in this part of ISO/IEC 10514

This part of ISO/IEC 10514 provides specifications for:

- required symbols for Modula-2 program representation, including comments, literals, and source code directives;
- the lexical structure, the syntactic structure and the semantics of Modula-2 programs, including programs that make use of system modules.
- the interface to and the semantics of standard Modula-2 library modules.
- those separate modules of the standard library that a conforming implementation is required to supply;
- violations of the rules for use of the language, system modules and standard library modules that a conforming implementation is required to detect;
- certain criteria for the size and complexity of programs that a conforming implementation must accept;
- further compliance requirements for implementations, including documentation requirements.

1.3 Specifications not within the scope of this part of ISO/IEC 10514

This part of ISO/IEC 10514 provides no specifications for:

- the underlying representation of predefined data types (except in the case of packedset types; see 7.1.7.1);

- the method by which implementations are invoked (including identification of the program module and associated definition and implementation modules);
- the method by which compilation modules are stored (including the correspondence between module names and system file names where files are used);
- the method by which implementations accept input (including the encoding of source text and including the number of compilation modules accepted for each invocation);
- performance aspects of implementations, and certain quality aspects not covered by 1.2;
- the effect of executing a program that uses extensions to the language, extensions to system modules or extensions to standard library modules, or that otherwise deviates from this part of ISO/IEC 10514;
- the effect of continuing execution of a program in which an exception has occurred and execution has continued without an exception being raised;
- the meaning of a program that relies on a definition of implementation-dependent values or implementation-dependent behaviour.

Section 2 : Normative References

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 10514. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this part of ISO/IEC 10514 are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO/IEC 10967-1:1994, *Information technology—Language independent arithmetic—Part 1: Integer and floating point arithmetic*.

ISO/IEC 13817-1:....¹⁾, *Information technology — Programming languages, their environments and system software interfaces — Vienna Development Method — Specification Language—Part 1: Base language*.

BS 6154:1981, *Method of defining Syntactic Metalanguage*.

IEC 559:1989, *Binary floating-point arithmetic for microprocessor systems*

¹⁾To be published.