

ISO/IEC 25436:2006-12 (E)

Information technology - Eiffel: Analysis, Design and Programming Language

Contents

Page

8.2.5	Definition: Terminal, non-terminal, token	24
8.2.6	Definition: Production	24
8.2.7	Kinds of production	25
8.2.8	Definition: Aggregate production	25
8.2.9	Definition: Choice production	25
8.2.10	Definition: Repetition production, separator	25
8.2.11	Basic syntax description rule	25
8.2.12	Definition: Non-production syntax rule	26
8.2.13	Textual conventions	26
8.2.14	Definition: Validity constraint	27
8.2.15	Definition: Valid	27
8.2.16	Validity: General Validity rule	27
8.2.17	Definition: Semantics	27
8.2.18	Definition: Execution terminology	27
8.2.19	Semantics: Case Insensitivity principle	27
8.2.20	Definition: Upper name, lower name	28
8.2.21	Syntax (non-production): Semicolon Optionality rule	28
8.3	The architecture of Eiffel software	29
8.3.1	Definition: Cluster, subcluster, contains directly, contains	29
8.3.2	Definition: Terminal cluster, internal cluster	29
8.3.3	Definition: Universe	29
8.3.4	Syntax: : Class names	30
8.3.5	Validity: Class Name rule	30
8.3.6	Semantics: Class name semantics	30
8.3.7	Definition: System, root type name, root procedure name	30
8.3.8	Definition: Type dependency	30
8.3.9	Validity: Root Type rule	31
8.3.10	Validity: Root Procedure rule	31
8.3.11	Definition: Root type, root procedure, root class	32
8.3.12	Semantics: System execution	32
8.4	Classes	32
8.4.1	Definition: Current class	32
8.4.2	Syntax: : Class declarations	32
8.4.3	Syntax: : Notes	32
8.4.4	Semantics: Notes semantics	33
8.4.5	Syntax: : Class headers	33
8.4.6	Validity: Class Header rule	33
8.4.7	Definition: Expanded, frozen, deferred, effective class	33
8.4.8	Syntax: : Obsolete marks	34
8.4.9	Semantics: Obsolete semantics	34
8.5	Features	34
8.5.1	Definition: Inherited, immediate; origin; redeclaration; introduce	34
8.5.2	Syntax: : Feature parts	35
8.5.3	Feature categories: overview	35
8.5.4	Syntax: : Feature declarations	35
8.5.5	Syntax: : New feature lists	36
8.5.6	Syntax: : Feature bodies	36
8.5.7	Validity: Feature Body rule	36
8.5.8	Validity: Definition: Variable attribute	37
8.5.10	Definition: Routine, function, procedure	37
8.5.11	Definition: Command, query	37
8.5.12	Definition: Signature, argument signature of a feature	27

8.5.13	Feature principle	38
8.5.14	Syntax: : Feature names	38
8.5.15	Syntax (non-production): Alias Syntax rule	38
8.5.16	Definition: Operator feature, bracket feature, identifier-only	39
8.5.17	Definition: Identifier of a feature name	39
8.5.18	Validity: Feature Identifier principle	39
8.5.19	Definition: Same feature name, same operator, same alias	39
8.5.20	Syntax: : Operators	40
8.5.21	Syntax: : Assigner marks	40
8.5.22	Validity: Assigner Command rule	40
8.5.23	Definition: Synonym	41
8.5.24	Definition: Unfolded form of a possibly multiple declaration	41
8.5.25	Validity: Feature Declaration rule	42
8.5.26	Validity: Alias Validity rule	43
8.6	The inheritance relation	43
8.6.1	Syntax: : Inheritance parts	43
8.6.2	Syntax (non-production): Feature adaptation rule	44
8.6.3	Definition: Parent part for a type, for a class	44
8.6.4	Validity: Class ANY rule	44
8.6.5	Validity: Universal Conformance principle	44
8.6.6	Definition: Unfolded inheritance part of a class	45
8.6.7	Definition: Multiple, single inheritance	45
8.6.8	Definition: Inherit, heir, parent	45
8.6.9	Definition: Conforming, non-conforming parent	45
8.6.10	Definition: Ancestor types of a type, of a class	45
8.6.11	Definition: Ancestor, descendant	46
8.6.12	Definition: Proper ancestor, proper descendant	46
8.6.13	Validity: Parent rule	46
8.6.14	Syntax: : Rename clauses	46
8.6.15	Validity: Rename Clause rule	47
8.6.16	Semantics: Renaming principle	47
8.6.17	Definition: Final name, extended final name, final name set	47
8.6.18	Definition: Inherited name	48
8.6.19	Definition: Declaration for a feature	48
8.7	Clients and exports	48
8.7.1	Definition: Client relation between classes and types	48
8.7.2	Definition: Client relation between classes	49
8.7.3	Definition: Supplier	49
8.7.4	Definition: Simple client	49
8.7.5	Definition: Expanded client	49
8.7.6	Definition: Generic client, generic supplier	49
8.7.7	Definition: Indirect client	49
8.7.8	Definition: Client set of a Clients part	49
8.7.9	Syntax: : Clients	50
8.7.10	Syntax: : Export adaptation	50
8.7.11	Validity: Export List rule	50
8.7.12	Definition: Client set of a feature	51
8.7.13	Definition: Available for call, available	51
8.7.14	Definition: Exported, selectively available, secret	51
8.7.15	Definition: Secret, public property	52
8.7.16	Definition: Incremental contract view, short form	52
8.7.17	Definition: Contract view, flat-short form	53
8.8	Routines	53
8.8.1	Definition: Formal argument, actual argument	53
8.8.2	Syntax: : Formal argument and entity declarations	55
8.8.3	Validity: Formal Argument rule	55
8.8.4	Validity: Entity Declaration rule	54
8.8.5	Syntax: : Routine bodies	54
8.8.6	Definition: Once routine, once procedure, once function	54
8.8.7	Syntax: : Local variable declarations	55
8.8.8	Validity: Local Variable rule	55
8.8.9	Definition: Local variable	55

8.8.10	Syntax: : Instructions	55
8.9	Correctness and contracts	56
8.9.1	Syntax: : Assertions	56
8.9.2	Syntax (non-production): Assertion Syntax rule	56
8.9.3	Definition: Precondition, postcondition, invariant	57
8.9.4	Definition: Contract, subcontract	57
8.9.5	Validity: Precondition Export rule	57
8.9.6	Validity: Definition: Availability of an assertion clause	57
8.9.7	Syntax: : "Old" postcondition expressions	58
8.9.8	Validity: Old Expression rule	58
8.9.9	Semantics: Old Expression Semantics, associated variable, associated exception marker	58
8.9.10	Semantics: Associated Variable Semantics	59
8.9.11	Syntax: : "Only" postcondition clauses	59
8.9.12	Validity: Only Clause rule	59
8.9.13	Definition: Unfolded feature list of an Only clause	60
8.9.14	Definition: Unfolded Only clause	60
8.9.15	Definition: Hoare triple notation (total correctness)	60
8.9.16	Semantics: Class consistency	60
8.9.17	Syntax: : Check instructions	61
8.9.18	Definition: Check-correct	61
8.9.19	Syntax: : Variants	61
8.9.20	Validity: Variant Expression rule	61
8.9.21	Definition: Loop invariant and variant	61
8.9.22	Definition: Loop-correct	61
8.9.23	Definition: Correctness (class)	62
8.9.24	Definition: Local unfolded form of an assertion	62
8.9.25	Semantics: Evaluation of an assertion	62
8.9.26	Semantics: Assertion monitoring	62
8.9.27	Semantics: Assertion violation	63
8.9.28	Semantics: Assertion semantics	63
8.9.29	Semantics: Assertion monitoring levels	63
8.10	Feature adaptation	63
8.10.1	Definition: Redeclare, redeclaration	64
8.10.2	Definition: Unfolded form of an assertion	64
8.10.3	Definition: Assertion extensions	64
8.10.4	Definition: Covariance-aware form of an inherited assertion	64
8.10.5	Definition: Combined precondition, postcondition	65
8.10.6	Definition: Inherited as effective, inherited as deferred	65
8.10.7	Definition: Effect, effecting	65
8.10.8	Definition: Redefine, redefinition	65
8.10.9	Definition: Name clash	66
8.10.10	Syntax: : Precursor	66
8.10.11	Definition: Relative unfolded form of a Precursor	66
8.10.12	Validity: Precursor rule	66
8.10.13	Definition: Unfolded form of a Precursor	67
8.10.14	Semantics: Precursor semantics	67
8.10.15	Syntax: : Redefinition	67
8.10.16	Validity: Redefine Subclause rule	67
8.10.17	Semantics: Redefinition semantics	67
8.10.18	Syntax: : Undefine clauses	68
8.10.19	Validity: Undefine Subclause rule	68
8.10.20	Semantics: Undefinition semantics	68
8.10.21	Definition: Effective, deferred feature	68
8.10.22	Definition: Effecting	68
8.10.23	Deferred Class property	68
8.10.24	Effective Class property	68
8.10.25	Definition: Origin, seed	69
8.10.26	Validity: Redeclaration rule	69
8.10.27	Definition: Precursor (joined features)	70
8.10.28	Validity: Join rule	70
8.10.29	Semantics: Join Semantics rule	71

8.11	Types	71
8.11.1	Syntax: : Types	71
8.11.2	Semantics: Direct instances and values of a type	72
8.11.3	Semantics: Instances of a type	72
8.11.4	Semantics: Instance principle	72
8.11.5	Definition: Instances, direct instances of a class	72
8.11.6	Base principle; base class, base type, based	72
8.11.7	Base rule	73
8.11.8	Validity: Class Type rule	73
8.11.9	Semantics: Type Semantics rule	73
8.11.10	Definition: Base class and base type of an expression	73
8.11.11	Semantics: Non-generic class type semantics	74
8.11.12	Definition: Expanded type, reference type	74
8.11.13	Definition: Basic type	74
8.11.14	Definition: Anchor, anchored type, anchored entity	74
8.11.15	Definition: Anchor set; cyclic anchor	74
8.11.16	Definition: Types and classes involved in a type	75
8.11.17	Definition: Deanchored form of a type	75
8.11.18	Validity: Anchored Type rule	75
8.11.19	Definition: Attached, detachable	76
8.11.20	Semantics: Attached type semantics	76
8.11.21	Definition: Stand-alone type	77
8.12	Genericity	77
8.12.1	Syntax: : Actual generic parameters	77
8.12.2	Syntax: : Formal generic parameters	77
8.12.3	Validity: Formal Generic rule	77
8.12.4	Definition: Generic class; constrained, unconstrained	78
8.12.5	Definition: Generic derivation, generic type, non-generic type	78
8.12.6	Definition: Self-initializing formal parameter	78
8.12.7	Definition: Constraint, constraining types of a Formal_generic	78
8.12.8	Syntax: : Generic constraints	78
8.12.9	Validity: Generic Constraint rule	79
8.12.10	Definition: Constraining creation features	79
8.12.11	Validity: Generic Derivation rule	79
8.12.12	Definition: Generic-creation-ready type	80
8.12.13	Semantics: Generically derived class type semantics	80
8.12.14	Definition: Base type of a single-constrained formal generic	80
8.12.15	Definition: Base type of an unconstrained formal generic	81
8.12.16	Definition: Reference or expanded status of a formal generic	81
8.12.17	Definition: Current type	81
8.12.18	Definition: Features of a type	81
8.12.19	Definition: Generic substitution	81
8.12.20	Generic Type Adaptation rule	81
8.12.21	Definition: Generically constrained feature name	82
8.12.22	Validity: Multiple Constraints rule	82
8.12.23	Definition: Base type of a multi-constraint formal generic type	82
8.13	Tuples	82
8.13.1	Syntax: : Tuple types	82
8.13.2	Syntax: : Manifest tuples	82
8.13.3	Definition: Type sequence of a tuple type	83
8.13.4	Definition: Value sequences associated with a tuple type	83
8.14	Conformance	83
8.14.1	Definition: Compatibility between types	83
8.14.2	Definition: Compatibility between expressions	84
8.14.3	Definition: Expression conformance	84
8.14.4	Validity: Signature conformance	84
8.14.5	Definition: Covariant argument	84
8.14.6	Validity: General conformance	85
8.14.7	Definition: Conformance path	85
8.14.8	Validity: Direct conformance: reference types	85
8.14.9	Validity: Direct conformance: formal generic	86
8.14.10	Validity: Direct conformance: expanded types	86

8.14.11	Validity: Direct conformance: tuple types	86
8.15	Convertibility	87
8.15.1	Definition: Conversion procedure, conversion type	87
8.15.2	Definition: Conversion query, conversion feature	87
8.15.3	Validity: Conversion principle	87
8.15.4	Validity: Conversion Asymmetry principle	87
8.15.5	Validity: Conversion Non-Transitivity principle	87
8.15.6	Syntax: : Converter clauses	87
8.15.7	Validity: Conversion Procedure rule	87
8.15.8	Validity: Conversion Query rule	88
8.15.9	Definition: Converting to a class	88
8.15.10	Definition: Converting to and from a type	88
8.15.11	Definition: Converting "through"	88
8.15.12	Semantics: Conversion semantics	89
8.15.13	Definition: Explicit conversion	89
8.15.14	Validity: Expression convertibility	89
8.15.15	Definition: Statically satisfied precondition	89
8.15.16	Validity: Precondition-free routine	90
8.16	Repeated inheritance	91
8.16.1	Definition: Repeated inheritance, ancestor, descendant	91
8.16.2	Semantics: Repeated Inheritance rule	91
8.16.3	Definition: Sharing, replication	92
8.16.4	Validity: Call Sharing rule	92
8.16.5	Semantics: Replication Semantics rule	92
8.16.6	Syntax: : Select clauses	92
8.16.7	Validity: Select Subclause rule	92
8.16.8	Definition: Version	92
8.16.9	Definition: Multiple versions	93
8.16.10	Validity: Repeated Inheritance Consistency constraint	93
8.16.11	Definition: Dynamic binding version	93
8.16.12	Definition: Inherited features	93
8.16.13	Semantics: Join-Sharing Reconciliation rule	93
8.16.14	Definition: Precursor	93
8.16.15	Validity: Feature Name rule	94
8.16.16	Validity: Name Clash rule	94
8.17	Control structures	95
8.17.1	Semantics: Compound (non-exception) semantics	95
8.17.2	Syntax: : Conditionals	95
8.17.3	Definition: Secondary part	95
8.17.4	Definition: Prevailing immediately	95
8.17.5	Semantics: Conditional semantics	95
8.17.6	Definition: Inspect expression	96
8.17.7	Syntax: : Multi-branch instructions	96
8.17.8	Definition: Interval	96
8.17.9	Definition: Unfolded form of a multi-branch	96
8.17.10	Definition: Unfolded form of an interval	96
8.17.11	Validity: Interval rule	97
8.17.12	Definition: Inspect values of a multi-branch	97
8.17.13	Validity: Multi-branch rule	97
8.17.14	Semantics: Matching branch	98
8.17.15	Semantics: Multi-Branch semantics	98
8.17.16	Syntax: : Loops	98
8.17.17	Semantics: Loop semantics	98
8.17.18	Syntax: : Debug instructions	98
8.17.19	Semantics: Debug semantics	98
8.18	Attributes	99
8.18.1	Syntax: : Attribute bodies	99
8.18.2	Validity: Manifest Constant rule	99
8.19	Objects, values and entities	100
8.19.1	Semantics: Type, generating type of an object; generator	100
8.19.2	Definition: Reference, void, attached, attached to	100
8.19.3	Semantics: Object principle	100

8.19.4	Definition: Object semantics	100
8.19.5	Definition: Non-basic class, non-basic type, field	100
8.19.6	Definition: Subobject, composite object	101
8.19.7	Definition: Entity, variable, read-only	101
8.19.8	Syntax: : Entities and variables	101
8.19.9	Validity: Entity rule	101
8.19.10	Validity: Variable rule	102
8.19.11	Definition: Self-initializing type	102
8.19.12	Semantics: Default Initialization rule	102
8.19.13	Definition: Self-initializing variable	103
8.19.14	Definition: Evaluation position, precedes	103
8.19.15	Definition: Setter instruction	104
8.19.16	Definition: Properly set variable	104
8.19.17	Validity: Variable Initialization rule	104
8.19.18	Definition: Variable setting and its value	105
8.19.19	Definition: Execution context	105
8.19.20	Semantics: Variable Semantics	105
8.19.21	Semantics: Entity Semantics rule	106
8.20	Creating objects	106
8.20.1	Semantics: Creation principle	106
8.20.2	Definition: Creation operation	107
8.20.3	Validity: Creation Precondition rule	107
8.20.4	Syntax: : Creators parts	107
8.20.5	Definition: Unfolded creators part of a class	107
8.20.6	Validity: Creation Clause rule	108
8.20.7	Definition: Creation procedures of a class	108
8.20.8	Creation procedure property	108
8.20.9	Definition: Creation procedures of a type	108
8.20.10	Definition: Available for creation; general creation procedure	109
8.20.11	Syntax: : Creation instructions	109
8.20.12	Definition: Creation target, creation type	109
8.20.13	Semantics: Creation Type theorem	109
8.20.14	Definition: Unfolded form of a creation instruction	109
8.20.15	Validity: Creation Instruction rule	109
8.20.16	Validity: Creation Instruction properties	110
8.20.17	Semantics: Creation Instruction Semantics	110
8.20.18	Syntax: : Creation expressions	111
8.20.19	Definition: Properties of a creation expression	111
8.20.20	Validity: Creation Expression rule	111
8.20.21	Validity: Creation Expression Properties	111
8.20.22	Semantics: Creation Expression Semantics	112
8.20.23	Definition: Garbage Collection, not enough memory available	112
8.21	Comparing and duplicating objects	112
8.21.1	Object comparison features from ANY	113
8.21.2	Syntax: : Equality expressions	113
8.21.3	Semantics: Equality Expression Semantics	113
8.21.4	Semantics: Inequality Expression Semantics	114
8.21.5	Copying and cloning features from ANY	114
8.21.6	Deep equality, copying and cloning	115
8.22	Attaching values to entities	115
8.22.1	Definition: Reattachment, source, target	115
8.22.2	Syntax: : Assignments	116
8.22.3	Validity: Assignment rule	116
8.22.4	Semantics: Reattachment principle	116
8.22.5	Semantics: Attaching an entity, attached entity	116
8.22.6	Semantics: Reattachment Semantics	116
8.22.7	Semantics: Assignment Semantics	117
8.22.8	Definition: Dynamic type	117
8.22.9	Definition: Polymorphic expression; dynamic type and class sets	117
8.22.10	Syntax: : Assigner calls	117
8.22.11	Validity: Assigner Call rule	118
8.22.12	Semantics: Assigner Call semantics	118

8.23	Feature call	118
8.23.1	Validity: Call Use rule	118
8.23.2	Syntax: : Feature calls	118
8.23.3	Syntax: : Actual arguments	119
8.23.4	Definition: Unqualified, qualified call	119
8.23.5	Definition: Target of a call	119
8.23.6	Definition: Target type of a call	119
8.23.7	Definition: Feature of a call	119
8.23.8	Definition: Imported form of a Non_object_call	120
8.23.9	Validity: Non-Object Call rule	120
8.23.10	Semantics: Non-Object Call Semantics	120
8.23.11	Validity: Export rule	120
8.23.12	Validity: Export Status principle	121
8.23.13	Validity: Argument rule	121
8.23.14	Validity: Target rule	121
8.23.15	Validity: Class-Level Call rule	122
8.23.16	Definition: Void-unsafe	122
8.23.17	Definition: Target Object	122
8.23.18	Semantics: Failed target evaluation of a void-unsafe system	122
8.23.19	Definition: Dynamic feature of a call	122
8.23.20	Definition: Freshness of a once routine call	122
8.23.21	Definition: Latest applicable target and result of a non-fresh call	123
8.23.22	Semantics: Once Routine Execution Semantics	123
8.23.23	Semantics: Current object, current routine	123
8.23.24	Semantics: Current Semantics	124
8.23.25	Semantics: Non-Once Routine Execution Semantics	124
8.23.26	Semantics: General Call Semantics	124
8.23.27	Definition: Type of a Call used as expression	124
8.23.28	Semantics: Call Result	125
8.23.29	Semantics: Value of a call expression	125
8.24	Eradicating void calls	125
8.24.1	Syntax: : Object test	125
8.24.2	Definition: Object-Test Local	126
8.24.3	Validity: Object Test rule	126
8.24.4	Definition: Conjunctive, disjunctive, implicative; Term, semistrict term	126
8.24.5	Definition: Scope of an Object-Test Local	126
8.24.6	Semantics: Object Test semantics	127
8.24.7	Semantics: Object-Test Local semantics	127
8.24.8	Definition: Read-only void test	127
8.24.9	Definition: Scope of a read-only void test	127
8.24.10	Definition: Certified Attachment Pattern	127
8.24.11	Definition: Attached expression	128
8.25	Typing-related properties	128
8.25.1	Definition: Catcall	128
8.25.2	Validity: Descendant Argument rule	129
8.25.3	Validity: Single-level Call rule	129
8.26	Exception handling	129
8.26.1	Definition: Failure, exception, trigger	129
8.26.2	Syntax: : Rescue clauses	130
8.26.3	Validity: Rescue clause rule	130
8.26.4	Validity: Retry rule	130
8.26.5	Definition: Exception-correct	130
8.26.6	Semantics: Default Rescue Original Semantics	130
8.26.7	Definition: Rescue block	130
8.26.8	Semantics: Exception Semantics	131
8.26.9	Definition: Type of an exception	131
8.26.10	Semantics: Exception Cases	131
8.26.11	Semantics: Exception Properties	132
8.26.12	Definition: Ignoring, continuing an exception	132
8.27	Agents	133
8.27.1	Definition: Operands of a call	133
8.27.2	Definition: Operand position	133

8.27.3	Definition: Construction time, call time	133
8.27.4	Syntactical forms for a call agent	133
8.27.5	Syntax: : Agents	134
8.27.6	Syntax: : Call agent bodies	134
8.27.7	Definition: Target type of a call agent	134
8.27.8	Validity: Call Agent rule	134
8.27.9	Definition: Associated feature of an inline agent	134
8.27.10	Validity: Inline Agent rule	134
8.27.11	Validity:Inline Agent Requirements	135
8.27.12	Definition: Call-agent equivalent of an inline agent	135
8.27.13	Semantics: Semantics of an inline agent	135
8.27.14	Semantics: Use of Result in an inline function agent	135
8.27.15	Definition: Open and closed operands	135
8.27.16	Definition: Open and closed operand positions	135
8.27.17	Definition: Type of an agent expression	135
8.27.18	Semantics: Agent Expression semantics	136
8.27.19	Semantics: Effect of executing call on an agent	136
8.28	Expressions	136
8.28.1	Syntax: : Expressions	136
8.28.2	Definition: Subexpression, operand	136
8.28.3	Semantics: Parenthesized Expression Semantics	137
8.28.4	Syntax: : Operator expressions	137
8.28.5	Operator precedence levels	137
8.28.6	Definition: Parenthesized Form of an expression	137
8.28.7	Definition: Target-converted form of a binary expression	138
8.28.8	Validity: Operator Expression rule	138
8.28.9	Semantics: Expression Semantics (strict case)	138
8.28.10	Definition: Semistrict operators	139
8.28.11	Semantics: Operator Expression Semantics (semistrict case)	139
8.28.12	Syntax: : Bracket expressions	139
8.28.13	Validity: Bracket Expression rule	139
8.28.14	Definition: Equivalent Dot Form of an expression	139
8.28.15	Validity: Boolean Expression rule	140
8.28.16	Validity: Identifier rule	140
8.28.17	Definition: Type of an expression	140
8.29	Constants	141
8.29.1	Syntax: : Constants	141
8.29.2	Validity: Constant Attribute rule	141
8.29.3	Syntax: : Manifest constants	141
8.29.4	Syntax (non-production): Sign Syntax rule	141
8.29.5	Syntax (non-production): Character Syntax rule	141
8.29.6	Definition: Type of a manifest constant	142
8.29.7	Validity: Manifest-Type Qualifier rule	142
8.29.8	Semantics: Manifest Constant Semantics	142
8.29.9	Definition: Manifest value of a constant	143
8.29.10	Syntax: : Manifest strings	143
8.29.11	Syntax (non-production): Line sequence	143
8.29.12	Syntax (non-production): Manifest String rule	144
8.29.13	Definition: Line_wrapping_part	144
8.29.14	Semantics: Manifest string semantics	144
8.29.15	Validity: Verbatim String rule	144
8.29.16	Semantics: Verbatim string semantics	144
8.29.17	Definition: Prefix, longest break prefix, left-aligned form	145
8.30	Basic types	145
8.30.1	Definition: Basic types and their sized variants	145
8.30.2	Definition: Sized variants of STRING	145
8.30.3	Semantics: Boolean value semantics	145
8.30.4	Semantics: Character types	145
8.30.5	Semantics: Integer types	146
8.30.6	Semantics: Floating-point types	146
8.30.7	Semantics: Address semantics	146
8.31	Interfacing with C, C++ and other environments	146

8.31.1	Syntax: : External routines	146
8.31.2	Semantics: Address semantics	147
8.31.3	Syntax: : Registered languages	147
8.31.4	Syntax: : External signatures	147
8.31.5	Validity: External Signature rule	147
8.31.6	Semantics: External signature semantics	148
8.31.7	Syntax: : External file use	148
8.31.8	Validity: External File rule	148
8.31.9	Semantics: External file semantics	149
8.31.10	Syntax: : C externals	149
8.31.11	Validity: C External rule	149
8.31.12	Semantics: C Inline semantics	150
8.31.13	Syntax: : C++ externals	150
8.31.14	Validity: C++ External rule	150
8.31.15	Semantics: C++ Inline semantics	150
8.31.16	Syntax: : DLL externals	150
8.31.17	Validity: External DLL rule	151
8.31.18	Semantics: External DLL semantics	151
8.32	Lexical components	151
8.32.1	Syntax (non-production): Character, character set	151
8.32.2	Definition: Letter, alpha_betic, numeric, alpha_numeric, printable	151
8.32.3	Definition: Break character, break	152
8.32.4	Semantics: Break semantics	152
8.32.5	Definition: Expected, free comment	152
8.32.6	Syntax (non-production): "Blanks or tabs", new line	152
8.32.7	Syntax: : Comments	153
8.32.8	Syntax (non-production): Free Comment rule	153
8.32.9	Semantics: Header Comment rule	153
8.32.10	Definition: Symbol, word	153
8.32.11	Syntax (non-production): Break rule	153
8.32.12	Semantics: Letter Case rule	154
8.32.13	Definition: Reserved word, keyword	154
8.32.14	Syntax (non-production): Double Reserved Word rule	154
8.32.15	Definition: Special symbol	154
8.32.16	Syntax (non-production): Identifier	154
8.32.17	Validity: Identifier rule	155
8.32.18	Definition: Predefined operator	155
8.32.19	Definition: Standard operator	155
8.32.20	Definition: Operator symbol	155
8.32.21	Definition: Free operator	155
8.32.22	Syntax (non-production): Manifest character	156
8.32.23	Special characters and their codes	156
8.32.24	Syntax (non-production): Percent variants	157
8.32.25	Semantics: Manifest character semantics	157
8.32.26	Syntax (non-production): String, simple string	157
8.32.27	Semantics: String semantics	157
8.32.28	Syntax: : Integers	157
8.32.29	Validity: Integer rule	158
8.32.30	Semantics: Integer semantics	158
8.32.31	Syntax (non-production): Real number	158
8.32.32	Semantics: Real semantics	159
Index	161