Differences between Modula-2 and Pascal

Hausi A. Muller

Department of Computer Science
Rice University
P.O. Box 1892
Houston, Texas 77251

1. Introduction

This paper outlines syntactical and semantical differences between Modula-2 and Pascal. Readers who know Pascal and intend to learn Modula-2 will benefit most from this document.

Modula-2 features which have no immediate correspondent in Pascal are not discussed in this paper. Such novelties include *modules, local modules, separate compilation, definition modules, implementation modules, import/export lists, opaque types* and *multiprogramming facilities.*

For a programmer's tutorial and a clear, concise report of Modula-2 refer to [Wirth 83].

2. Syntactic sugar

Feature	Modula-2	Pascal
case of reserved words	upper case; WHILE	case is not significant; while = WHILE
case of identifiers	case is significant; IsEmpty <> isempty	case is not significant; IsEmpty = isempty
number of significant characters in identifiers	all characters are significant	a fixed number of characters is significant
character constants	'@', "@"	'@'
strings	"That's incredible!" 'Codeword "Barbarossa"'	'That''s incredible!' 'Codeword "Barbarossa"'
single and double quotes in one string	cannot be done	'"That''s incredible!"'
comments	(* *); may be nested	{ } (* *); must not be nested
set brackets	{}	[]
constant NIL	standard identifier	reserved word

function keyword	PROCEDURE	FUNCTION
parameterless function declaration	PROCEDURE p(): CHAR ;	FUNCTION p: CHAR ;
sequence of declarations in a block; the symbols [], {}, | are meta-symbols of EBNF (Extended Backus-Naur Form, p. 10, [Wirth 83]); angular brackets [] denote optionality of the enclosed sentential form; curly brackets {} denote its repetition (possibly zero times).	{ ConstDec | TypeDec | VarDec | ProcDec }	[ConstDec] [TypeDec] [VarDec] { ProcDec }
	CONST N = 32 ;	CONST N = 32 ; N1 = 31 ; N2 = 63 ;
	TYPE index = [0..N-1] ; buf = ARRAY index OF CHAR ;	TYPE index = 0..N1 ; buf = ARRAY [index] OF CHAR ; entry = RECORD a: CHAR ; b: 0..MaxCard END ; sequence = ARRAY [0..N2] OF CHAR ;
	VAR b: buf ;	
	PROCEDURE BufHandler ; END BufHandler ;	PROCEDURE BufHandler ; END ; (* BufHandler *)
	TYPE entry = RECORD a: CHAR ; b: CARDINAL END ; sequence = ARRAY [0..2*N-1] OF entry ;	PROCEDURE EnterEntry ; END ; (* EnterEntry *)
	VAR s: sequence ;	
	PROCEDURE EnterEntry ; END EnterEntry ;	
storage allocation	FROM Storage IMPORT ALLOCATE ; NEW(x)	NEW(x)
storage deallocation	FROM Storage IMPORT DEALLOCATE ; DISPOSE(x)	DISPOSE(x)
separator in variant records and case statements	|	;
repeating procedure identifier	PROCEDURE BufHandler ; END BufHandler ;	PROCEDURE BufHandler ; END ; (* BufHandler *)

indirect recursion

```
PROCEDURE a ;
BEGIN
 b
END a ;

PROCEDURE b ;
BEGIN
 a
END b ;
```

```
PROCEDURE b ; FORWARD ;

PROCEDURE a ;
BEGIN
 b
END ;  (* a *)

PROCEDURE b ;
BEGIN
 a
END ;  (* b *)
```

3. Types

Feature	Modula-2	Pascal
subrange type	index = [0..31]	index = 0..31
unsigned integers	standard type CARDINAL	cardinal = 0..MaxCard
pointer type	x = POINTER TO t	x = ^t
bit access type	standard type BITSET	PACKED SET OF 0..WordLengthMinusOne
procedure type	a procedure can be an object like a variable of any other type	procedure parameters only
parameterless procedure type	PROC	not defined
variant records (semicolon, parenthesis, explicit END)	no restrictions; same syntax as case statements; RECORD CASE BOOLEAN OF TRUE: u,v: INTEGER | FALSE: r,s: CHAR END END	explicit variant must be the last record entry; RECORD CASE BOOLEAN OF TRUE: (u,v: INTEGER) ; FALSE: (r,s: CHAR) END
array declaration	ARRAY [1..3],['a'..'z'] OF INTEGER	ARRAY [1..3,'a'..'z'] OF INTEGER
string type	ARRAY [1..N] OF CHAR	PACKED ARRAY [1..N] OF CHAR
packing	not defined	PACKED StructuredType

unbounded array parameters, arrays of varying length	open arrays; PROCEDURE p(a: ARRAY OF CHAR)	implementation dependent
low-level storage unit type	WORD; to be imported from module SYSTEM; compatible with CARDINAL, INTEGER, BITSET, pointers	variant records
address manipulation type	ADDRESS; to be imported from module SYSTEM; compatible with CARDINAL; cardinal arithmetic	address = RECORD CASE BOOLEAN OF TRUE: (p: pointer) ; FALSE: (a: INTEGER) END

4. Statements

Feature	Modula-2	Pascal
statement terminator	each statement has an explicit terminating symbol; UNTIL for the RepeatStatement and END the rest; no compound statement	statement or SimpleStatement
BITSET assignment	x := {3}	x := [3]
arbitrary set assignment	y := TypeId{A,B}; TypeId is CHAR, INTEGER, CARDINAL, enumeration, subrange type ident	y := [A,B]

ReturnStatement	PROCEDURE p(): CHAR ;
	BEGIN
	StatementList ;
	IF b THEN
	RETURN('@')
	END ;
	StatementList ;
	RETURN('!')
	END p ;
	
	ReturnStatements can also appear in procedures and modules without expression.
		FUNCTION p: CHAR ;
		LABEL 1 ;
		BEGIN
		StatementList ;
		IF b THEN
		BEGIN
		p := '@' ;
		GOTO 1
		END ;
		StatementList ;
		p := '!' ;
		1:
		END ; (* p *)

IfStatement

```
IF b1 THEN
 a := 3
ELSIF b2 THEN
 a := 4
ELSE
 a := 5 ;  c := 7
END
```

```
IF b1 THEN a := 3 ELSE
BEGIN
 IF b2 THEN a := 4 ELSE
 BEGIN
  a := 5 ;  c := 7
 END
END
```

CaseStatement
(subrange,
expression,
else clause,
| as separator)

```
CASE i OF
 2:      StatementList1 |
 3..5:   StatementList2 |
 2*3:    StatementList3
 ELSE  StatementList4
END
```

```
IF (i>0) AND (i<5) THEN BEGIN
 CASE i OF
  2: BEGIN StatementList1 END ;
  3: BEGIN StatementList2 END ;
  4: BEGIN StatementList2 END ;
  5: BEGIN StatementList2 END ;
  6: BEGIN StatementList3 END
 END  (* case *)
END ELSE BEGIN
 StatementList4
END  (* if *)
```

ForStatement

```
FOR i := 1 TO 3 DO
 StatementList
END
```

```
FOR i := 1 TO 3 DO BEGIN
StatementList
END
```

```
FOR i := 9 TO 1 BY -1 DO
 StatementList
END
```

```
FOR i := 9 DOWNTO 1 DO BEGIN
StatementList
END
```

```
FOR i := 0 TO 55 BY 5 DO
 StatementList
END
```

```
i := 0 ;
WHILE i <= 55 DO BEGIN
 StatementList ;
 i := i + 5
END
```

LoopStatement

```
LOOP
 StatementList
END
```

```
WHILE TRUE DO BEGIN
 StatementList
END
```

ExitStatement

```
LOOP
 StatementList1 ;
 IF b THEN EXIT END ;
 StatementList2
END
```

```
WHILE TRUE DO BEGIN
 StatementList1 ;
 IF b THEN GOTO 1 ;
 StatementList2
END ;
1:
```

GotoStatement

not defined

```
LABEL 1 ;
GOTO 1 ;
1:
```

5. Expressions and standard procedures

Feature	Modula-2	Pascal
evaluation of expressions	"short-circuit"; the AND and OR operators skip the second operand if the expression value can be detected from the first operand. IF (p<>NIL) AND (p^.key<>x) THEN StatementList END	no evaluation order can be assumed IF p <> nil THEN BEGIN IF p^.key <> x THEN BEGIN StatementList END END
constant declarations	constant expressions N = 100 ; limit = 2*N-1 ;	not defined N = 100 ; limit = 199 ;
case labels	constant expressions	not defined
scale factor	3.0E+12	3.0E+12 ; 3.0e+12
fast increments and decrements	INC(i) INC(i,d) DEC(i) DEC(i,d)	i := i + 1 i := i + d i := i - 1 i := i - d
predecessor	DEC(i)	PRED(i)
successor	INC(i)	SUCC(i)
inverse of ORD	VAL(TypeId,ORD(x)) = x; TypeId is CHAR, INTEGER, CARDINAL, enumeration, subrange type ident	not defined, except for CHAR, CHR(x)
round to cardinal	TRUNC(x+0.5)	ROUND(x)
arithmetic functions	library module	standard functions
low index bound of unbounded array	always equal to 0	implementation dependent
high index bound of unbounded array	HIGH(a)	implementation dependent

terminate program execution	HALT	LABEL 99 ; BEGIN ... GOTO 99 ; ... 99: END.
capitalize character	CAP(ch)	IF ch IN ['a'..'z'] THEN ch := CHR(ORD(ch) - ORD('a') + ORD('A'))
symmetric set difference	A / B	(A-B) + (B-A)
set inclusion	INCL(S,i)	S := S + [i]
set exclusion	EXCL(S,i)	S := S - [i]
not equal	<> #	<>
logical and	AND &	AND
address of a variable x	ADR(x); to be imported from module SYSTEM	not defined
number of storage units assigned to variable x	SIZE(x); to be imported from module SYSTEM	not defined
number of storage units assigned to variable of type t	TSIZE(t); to be imported from module SYSTEM	not defined
cardinal to real conversion	FLOAT(x)	implicit conversion
type transfer functions; representation is not changed	variables of type WORD, CARDINAL, INTEGER, REAL, BITSET, ADDRESS, and pointers can be transferred into each other; x := TypeId(y); TypeId is one of the above; x is a variable of this type; y is a variable of one of the above types	using variant records

6. Miscellaneous facilities

Feature	Modula-2	Pascal
input and output facilities	not defined in the language; library modules;	defined in the language
compilation units	main MODULE DEFINITION MODULE IMPLEMENTATION MODULE	implementation dependent
static variables	variables at a module level	one set of global variables only
dynamic storage allocation scheme	can be explicitly programmed	implicit; cannot be altered

References

[Digi 82] Digital Equipment Corporation, VAX-11 Pascal Reference Manual, V2.0, Order No. AA-H484C-TE, October 1982

[EKMP 83] Eckhardt, H., Koch, J., Mall, M., Putfarken, P., Universitaet Hamburg, VAX-11 Modula-2 User's Guide, April 1983

[Eldr 84] Eldred, Eric, Massachusetts General Hospital, Boston, "Volition Systems' Modula-2: A Version of Modula-2 for the Apple II," *Byte Publications Inc.*, June 1984

[IEEE 83] *American National Standard Pascal Computer Programming Language*, ANSI/IEEE770X3.97-1983, IEEE Publications, January 1983

[JeWi 78] Jensen, Kathleen, Wirth, Niklaus, ETH Zurich, *Pascal User Manual and Report*, Second Edition, Springer-Verlag, 1978

[McCo 83] Joel McCormack, Richard Gleaves, Volition Systems, "Modula-2 A Worthy Successor to Pascal," *Byte Publications Inc.*, April 1983

[Spec 83] Spector, David, Prime Computer, Inc., "Ambiguities and Insecurities in Modula-2," *SIGPLAN Notices*, Vol. 17, No. 8, 1982

[SuGl 83] Sumner, Roger T., Gleaves, R. E., Volition Systems, "Modula-2 -- A Solution to Pascal's Problems," *SIGPLAN Notices*, Vol. 17, No. 9, 1982

[Wirth 83] Wirth, Niklaus, ETH Zurich, *Programming in Modula-2*, Second Edition, Springer-Verlag, 1983