```
(* Test Cases for the (Pygments) Modula-2 Lexer *)

(* Notes:
   (1) Without dialect option nor embedded dialect tag, the lexer operates in
       fallback mode, recognising the *combined* literals, punctuation symbols
       and operators of all supported dialects, and the *combined* reserved
       words and builtins of PIM Modula-2, ISO Modula-2 and Modula-2 R10.
   (1) If multiple embedded dialect tags are present, the lexer will use the
       first valid tag and ignore any subsequent dialect tags in the file.
   (2) An embedded dialect tag overrides any command line dialect option. *)


(* Testing command line dialect option *)

(* to select a dialect via command line, a dialect option may be passed:

    $ pygmentize -O full,dialect=specifier ...

    where specifier is defined by the EBNF below:

    specifier :
      baseDialect ( "+" languageExtension )? ;

    baseDialect :
      m2pim | m2iso | m2r10 | objm2 ;

    languageExtension :
      gm2 | mocka | aglet | gpm | p1 | sbu | xds ;

    whereby valid extensions for m2pim are gm2 and mocka,
    and valid extensions for m2iso are gm2, aglet, p1, sbu and xds.

    Example: $ pygmentize -O full,dialect=m2iso+gm2 -f html -o outfile infile *)


(* Testing embedded dialect tags *)

(* to activate a dialect tag, remove whitespace before the exclamation mark *)

(* !m2pim*)       (* for PIM Modula-2 *)
(* !m2iso*)       (* for ISO Modula-2 *)
(* !m2r10*)       (* for Modula-2 R10 *)
(* !objm2*)       (* for Objective Modula-2 *)
(* !m2pim+gm2*)   (* for PIM Modula-2 with GNU extensions *)
(* !m2pim+mocka*) (* for PIM Modula-2 with MOCKA extensions *)
(* !m2iso+aglet*) (* for ISO Modula-2 with Aglet extensions *)
(* !m2iso+gm2*)   (* for ISO Modula-2 with GNU extensions *)
(* !m2iso+gpm*)   (* for ISO Modula-2 with Gardens Point extensions *)
(* !m2iso+p1*)    (* for ISO Modula-2 with p1 extensions *)
(* !m2iso+sbu*)   (* for ISO Modula-2 with Stony Brook extensions *)
(* !m2iso+xds*)   (* for ISO Modula-2 with XDS extensions *)


(* Test Report *)

(* Selected Dialect: ISO Modula-2 with p1 Extensions *)

(* Rendered in Algol Publication Mode: True *)


(* At a Glance *)

(* highlighted if recognised *)

export (* PIM and ISO *)              packedset (* ISO *)

ARGLIST (* M2 R10 and ObjM2 *)        BYCOPY (* ObjM2 only *)

__FILE__ (* GNU only *)               FOREIGN (* MOCKA and GPM *)

BITSET8 (* Aglet, GNU and M2 R10 *)   bcd (* p1 and M2 R10 *)

FUNC (* Stony Brook only *)           SEQ (* XDS only *)
```

```
(* Literal Tests *)
(* Number Literals *)
(* non-affixed, whole numbers, supported by all dialects *)
(* rendered as number literals *)
n := 123; m := 1000000;

(* non-affixed, real numbers, no exponent, supported by all dialects *)
(* rendered as number literals *)
r1 := 123.45; r2 := 1000000.00;

(* non-affixed, real numbers, exponent with 'e', supported by all dialects *)
(* rendered as number literals *)
r1 := 1.2345e+6; r2 := 1.2345e-6; r3 := 1.2345e1000;

(* non-affixed, real numbers, exponent with 'E', supported by PIM + ISO *)
(* rendered as number literals if supported, as errors if unsupported *)
r1 := 1.2345₁₀+6; r2 := 1.2345₁₀-6; r3 := 1.2345₁₀1000;

(* non-affixed, real numbers, digit separators, supported by M2 R10 + ObjM2 *)
(* rendered as number literals if supported, as errors if unsupported *)
r1 := 1.234'567'8e+9; r2 := 1.234'567'8e-9; r3 := 1.234'567'89e1'000;

(* prefixed, base-2 and base-16, supported by M2 R10 + ObjM2 *)
(* rendered as number literals if supported, as errors if unsupported *)
base2 := 0b0110; base2:= 0b1100'1100'0110'1001;
base16 := 0xFF00; base16:= 0xFF00'0000'0000'0000;

(* suffixed, base-8 and base-16. supported by PIM + ISO *)
(* rendered as number literals if supported, as errors if unsupported *)
(* B *) base8 := 377₈;   (* H *) base16 := 0FF00000000000000₁₆;

(* suffixed, real numbers, supported by p1 *)
(* rendered as number literals if supported, as errors if unsupported *)
bcd := 999.00$;

(* Numeric Character Literals *)
(* prefixed, supported by M2 R10 + ObjM2 *)
(* rendered as number literals if supported, as errors if unsupported *)
char := 0u20; unichar := 0u2038
```

```
(* suffixed, supported by PIM + ISO *)

(* rendered as number literals if supported, as errors if unsupported *)

char := 377C;


(* Quoted Character Literals *)

(* supported by all dialects *)

(* rendered as strings *)

char := 'a'; char := "a";


(* String Literals *)

(* supported by all dialects *)

(* rendered as strings *)

string := 'The cat said "meow!".';
string := "It is eight O'clock.";


(* Punctuation Tests *)

(* Common Punctuation *)

(* supported by all dialects *)

(* . *) Foo.Bar.Baz;

(* , *) foo, bar, baz;

(* ; *) foo; bar; baz;

(* : *) var i : integer;

(* | *) case foo of | 0 : bar | 1 : baz | 2 : bam end;

(* := *) foo := bar;

(* .. *) type Sign = [-1..1] of integer;

(* () *) foo(bar);

(* [] *) array[n];

(* {} *) const Foo = { 1, 2, 3 };


(* Synonym for Vertical Bar *)

(* supported by ISO *)

(* rendered as errors if unsupported *)

(* ! *) case foo of 0 : bar ! 1 : baz ! 2 : bam end;


(* Ascend/Descend and Increment/Decrement Suffix *)

(* supported by M2 R10 + ObjM2 *)

(* rendered as errors if unsupported *)

(* ++ *) for key++ in collection do collection[key]++ end;

(* -- *) for index-- in range do array[index]-- end;
```

```
(* Wildcard Import *)

(* supported by M2 R10, ObjM2, p1 and Stony Brook *)

(* rendered as error if unsupported *)

from ASSEMBLER import *;


(* Import to Re-Export *)

(* supported by M2 R10 + ObjM2 *)

(* rendered as error if unsupported *)

import Foo+, Bar+;


(* Generic Parameter Delimiters *)

(* supported by M2 R10 + ObjM2 *)

(* rendered as errors if unsupported *)

(* << >> *) GENLIB Foo from Template for Bar = <<array of char>> end;


(* Blueprint Punctuation *)

(* supported by M2 R10 + ObjM2 *)

(* rendered as errors if unsupported *)

(* Undetermined Type Property Suffix ? *)

TPROPERTIES = isCollection, isIndexed | isRigid?;

(* Ancillary Blueprint Constant Prefix ~ *)

const ~ isFoobar = Foo and Bar;

(* Blueprint Requirement Implication -> *)

isFoobar ->
procedure [abs] abs ( i : ProtoInteger ) : ProtoInteger;

(* Unary Minus Binding symbol *)

procedure [+/-] unaryMinus ( i : ProtoInteger ) : ProtoInteger;

(* Binding Selector # *)

procedure [STORE|#] storeValues ...;

(* Binding Selector * *)

procedure [REMOVE|*] removeAllValues;
  ( target : ProtoArray; fromIndex : IndexType; values : array of ValueType );


(* Non-Alphabetic Operator Tests *)

(* Common Operators *)

(* supported by all dialects *)

(* +, -, *, / *) a := b + c - d * e / f;

(* =, <, > *) equal := a = b; less := a < b; greater := a > b;

(* #, >=, <= *) notEqual := a ≠ b; notLess := a ≥ b; notGreater := a ≤ b;

(* ^ *) next := this^.next
```

```
(* Pascal Style Inequality Operator *)
(* supported by PIM + ISO *)
(* rendered as error if unsupported *)
(* <> *) bool := a <> b;

(* Synonyms for NOT and AND Operators *)
(* supported by PIM + ISO *)
(* rendered as errors if unsupported *)
(* ~ *) not := ¬ a;
(* & *) and := a & b;

(* Synonym for Pointer Dereferencing Operator *)
(* supported by ISO *)
(* rendered as error if unsupported *)
(* @ *) next := this@.next;

(* ISO 80000-2 Compliant Set Difference Operator *)
(* supported by M2 R10 + ObjM2 *)
(* rendered as error if unsupported *)
(* \ *) setDifference := A \ B;

(* Identity Test Operator *)
(* supported by M2 R10 + ObjM2 *)
(* rendered as error if unsupported *)
(* == *) identical := a ≡ b;

(* Type Conversion Operator *)
(* supported by M2 R10 + ObjM2 *)
(* rendered as error if unsupported *)
(* :: *) int := real :: integer;

(* Dot Product Operator *)
(* supported by M2 R10 + ObjM2 *)
(* rendered as error if unsupported *)
(* *. *) dotProduct := v1 • v2;

(* Array Concatenator *)
(* supported by M2 R10 + ObjM2 *)
(* rendered as error if unsupported *)
```

```
(* +> *) concatenation := array1 +> array2;

(* Smalltalk Message Prefix *)
(* supported by ObjM2 *)
(* rendered as error if unsupported *)
(* ` *) str := `NSString alloc init;

(* Single Line Comment Tests *)
(* Ada Style Comment *)
(* supported by XDS Modula-2 Extensions *)
(* rendered as comment if supported, as error if unsupported *)
(* at start of line *)
-- Bamboo
(* not at start of line *)
foo := bar(baz); -- Bamboo

(* BCPL Style Comment *)
(* permitted by M2 R10 specification as a compiler option for Doxygen support *)
(* rendered as comment if supported, as error if unsupported *)
(* at start of line *)
// Bamboo
/// Doxygen tag
//! Qt Style Doxygen tag
(* not at start of line *)
(* not permitted by any dialect *)
foo := bar(baz); // Bamboo

(* Fortran Style Comment *)
(* permitted by M2 R10 specification as a compiler option for Doxygen support *)
(* rendered as comment if supported, as error if unsupported *)
(* at start of line *)
! Bamboo
!< Start of Doxygen tag
!!
!> End of Doxygen tag
(* not at start of line *)
foo := bar(baz); ! Bamboo

(* Pragma Delimiter Tests *)
(* PIM style pragma *)
```

```
(* rendered as pragma if supported, as comment if unsupported *)
(*$INLINE*) (* PIM *)

(* ISO style pragma *)
(* rendered as pragma if supported, as error if unsupported *)
<*INLINE*>  (* dialects other than PIM *)

(* Stony Brook pragma *)
(* rendered as pragma if supported, as error if unsupported *)
%IF foo %THEN bar %ELSE baz %END

(* Substitution Tests When in Algol Mode *)
(* Number Literal Suffix Substitutions *)
(* number literal suffixes are replaced with subscript numeric suffixes *)
(* suffix B *)
base8 := 377₈;
(* suffix H *)
base16 := 0FF00000000000000₁₆;

(* Exponent Indicator Substitution *)
(* the uppercase E exponent indicator is replaced with subscript 10 *)
r := 1.2345678₁₀9;

(* Math Symbol Substitution *)
(* the following operators are replaced with proper math symbols *)
(* not-equal operator # *)
if foo ≠ bar then ... end;
(* not-less operator >= *)
if foo ≥ bar then ... end;
(* not-greater operator <= *)
if foo ≤ bar then ... end;
(* identity operator == *)
if foo ≡ bar then ... end;
(* dot product operator *. *)
dotProduct := v1 • v2;
(* NOT operator synonym ~ *)
not := ¬ a;

(* Identifier Tests *)
(* Alpha-Numeric Identifiers *)
```

```
(* recognised by all dialects *)
(* rendered as identifiers if recognised, as errors if not recognised *)
lowercase123 camelCase123 TitleCase123 UPPERCASE123

(* Alpha-Numeric Identifiers with Lowline *)
(* recognised by all ISO dialects, M2 R10 and ObjM2 *)
(* rendered as identifiers if recognised, as errors if not recognised *)
_lowercase123 _camelCase123 _TitleCase123 _UPPERCASE123
lowercase_123 camelCase_123 TitleCase_123 UPPERCASE_123
lowercase123_ camelCase123_ TitleCase123_ UPPERCASE123_

(* POSIX Identifiers *)
(* recognised by M2 R10, ObjM2, p1 and GM2 *)
(* rendered as identifiers if recognised, as errors if not recognised *)
$lowercase_123 $camelCase_123 $TitleCase_123 $UPPERCASE_123
lowercase_$123 camelCase_$123 TitleCase_$123 UPPERCASE_$123
lowercase_123$ camelCase_123$ TitleCase_123$ UPPERCASE_123$

(* OpenVMS Identifiers *)
(* permitted implementation option for M2 R10 and ObjM2 *)
(* rendered as identifiers if recognised, as errors if not recognised *)
%lowercase_$123 %camelCase_$123 %TitleCase_$123 %UPPERCASE_$123
lowercase_$%123 camelCase_$%123 TitleCase_$%123 UPPERCASE_$%123
lowercase_$123% camelCase_$123% TitleCase_$123% UPPERCASE_$123%

(* M2 R10 Template Engine Placeholders *)
(* supported by M2 R10 and ObjM2 *)
(* rendered as placeholder if supported, as error if unsupported *)
definition module ##CustomInteger## [ProtoInteger];

type ##CustomInteger## = record
  * hidden : array ##BitwidthInOctets## of OCTET
end;

(* Reserved Words Tests *)
(* Common Reserved Words *)
(* supported by all dialects *)
(* rendered as keywords *)
and array begin by case const definition div do else elsif end exit for from
if implementation import in loop mod module not of or pointer procedure
record repeat return set then to type until var while
```

```
(* Additional Reserved Words for PIM *)

(* recognised by PIM dialects *)

(* rendered as keywords if recognised, as identifiers if not recognised *)

export qualified with


(* Additional Reserved Words for ISO *)

(* recognised by ISO dialects *)

(* rendered as keywords if recognised, as identifiers if not recognised *)

(* ISO 10514-1 *)

except export finally forward packedset qualified rem retry with

(* ISO 10514-2 & ISO 10514-3 *)

abstract as class guard inherit override readonly reveal traced unsafeguarded


(* Additional Reserved Words for M2 R10 *)

(* recognised by M2 R10 and ObjM2 *)

(* rendered as keywords if recognised, as identifiers if not recognised *)

(* core language *)

ALIAS ARGLIST BLUEPRINT COPY GENLIB INDETERMINATE new NONE OPAQUE REFERENTIAL
RELEASE RETAIN

(* with symbolic assembler option *)

ASM REG


(* Additional Reserved Words for ObjM2 *)

(* recognised by ObjM2 *)

(* rendered as keywords if recognised, as identifiers if not recognised *)

BYCOPY BYREF class CONTINUE CRITICAL INOUT METHOD ON OPTIONAL OUT PRIVATE
PROTECTED PROTOCOL PUBLIC SUPER TRY


(* Additional Reserved Words for GNU Extensions to PIM and ISO *)

(* recognised by GM2 *)

(* rendered as keywords if recognised, as identifiers if not recognised *)

ASM __ATTRIBUTE__ __BUILTIN__ __COLUMN__ __DATE__ __FILE__ __FUNCTION__
__LINE__ __MODULE__ VOLATILE


(* Additional Reserved Words for MOCKA Extensions to PIM *)

(* recognised by MOCKA *)

(* rendered as keyword if recognised, as identifier if not recognised *)

FOREIGN


(* Additional Reserved Words for Gardens Point Extensions to ISO *)

(* recognised by Gardens Point *)
```

```
(* rendered as keyword if recognised, as identifier if not recognised *)

FOREIGN


(* Additional Reserved Words for Stony Brook Extensions to ISO *)

(* recognised by Stony Brook *)

(* rendered as keyword if recognised, as identifier if not recognised *)

BIG BITFIELDS BREAK CONTINUE except FUNC INOUT OUT MACRO SMALL


(* Additional Reserved Words for XDS Extensions to ISO *)

(* recognised by XDS *)

(* rendered as keyword if recognised, as identifier if not recognised *)

SEQ


(* Builtins Tests *)

(* Common Builtins *)

(* recognised by all dialects *)

(* rendered as builtins if recognised, as identifiers if not recognised *)

(* proper builtins *)

abs boolean cardinal char chr false integer longint longreal
max min nil odd ord real true

(* pseudo-builtins *)

address word adr


(* Additional Builtins for PIM *)

(* recognised by PIM dialects *)

(* rendered as builtins if recognised, as identifiers if not recognised,
   NEW is rendered as a keyword if the dialect is M2 R10, ObjM2 or unknown *)

(* proper builtins *)

bitset cap dec dispose excl float halt high inc incl new nil proc size trunc val

(* pseudo-builtins *)

system PROCESS tsize NEWPROCESS transfer


(* Additional Builtins for ISO *)

(* recognised by ISO dialects *)

(* rendered as builtins if recognised, as identifiers if not recognised *)

(* proper builtins *)

(* ISO 10514-1 *)

bitset cap cmplx complex dec dispose excl float halt high im inc incl int
interruptible length lfloat longcomplex new proc protection re size trunc
uninterrubtible val

(* ISO 10514-2 & ISO 10514-3 *)
```

```
create destroy empty ismember self

(* pseudo-builtins *)

(* SYSTEM pseudo-module *)

system bitsperloc byte locsperbyte locsperword loc addadr subadr difadr makeadr
adr rotate shift cast tsize

(* COROUTINES pseudo-module *)

coroutines attach coroutine current detach handler interruptsource iotransfer
IsATTACHED listen newcoroutine prot transfer

(* EXCEPTIONS pseudo-module *)

exceptions AllocateSource CurrentNumber ExceptionNumber ExceptionSource
GetMessage IsCurrentSource IsExceptionalExecution raise

(* TERMINATION pseudo-module *)

termination IsTerminating HasHalted

(* M2EXCEPTION pseudo-module *)

m2exception M2Exceptions M2Exception IsM2Exception indexException rangeException
caseSelectException invalidLocation functionException wholeValueException
wholeDivException realValueException realDivException complexValueException
complexDivException protException sysException coException exException


(* Additional Builtins for M2 R10 *)

(* recognised by M2 R10 and ObjM2 *)

(* rendered as builtins if recognised, as identifiers if not recognised *)

(* proper builtins *)

cardinal COUNT empty EXISTS INSERT length LONGCARD OCTET PTR PRED READ READNEW
REMOVE RETRIEVE SORT STORE SUBSET SUCC TLIMIT TMAX TMIN true tsize UNICHAR
WRITE WRITEF

(* pseudo-builtins *)

(* TPROPERTIES builtin-module *)

TPROPERTIES PROPERTY LITERAL TPROPERTY TLITERAL TBUILTIN TDYN TREFC TNIL
TBASE TPRECISION TMAXEXP TMINEXP

(* CONVERSION builtin-module *)

CONVERSION TSXFSIZE SXF val

(* UNSAFE builtin-module *)

UNSAFE cast byte INTRINSIC AVAIL ADD SUB ADDC SUBC FETCHADD FETCHSUB SHL SHR
ASHR ROTL ROTR ROTLC ROTRC BWNOT BWAND BWOR BWXOR BWNAND BWNOR SETBIT TESTBIT
LSBIT MSBIT CSBITS BAIL halt TODO FFI ADDR VARGLIST VARGC

(* ATOMIC builtin-module *)

ATOMIC INTRINSIC AVAIL SWAP CAS inc dec BWAND BWNAND BWOR BWXOR

(* COMPILER builtin-module *)

COMPILER DEBUG MODNAME PROCNAME LINENUM DEFAULT HASH

(* ASSEMBLER builtin-module *)

ASSEMBLER REGISTER SETREG GETREG CODE
```

```
(* Identifiers of first class ADTs for M2 R10 *)

(* provided by M2 R10 standard library *)

(* rendered as builtins when dialect is set to Modula-2 R10,
   this can be turned off by option treat_stdlib_adts_as_builtins=off *)

bcd LONGBCD bitset SHORTBITSET LONGBITSET LONGLONGBITSET complex longcomplex
SHORTCARD LONGLONGCARD SHORTINT LONGLONGINT POSINT SHORTPOSINT LONGPOSINT
LONGLONGPOSINT BITSET8 BITSET16 BITSET32 BITSET64 BITSET128 BS8 BS16 BS32
BS64 BS128 CARDINAL8 CARDINAL16 CARDINAL32 CARDINAL64 CARDINAL128 CARD8
CARD16 CARD32 CARD64 CARD128 INTEGER8 INTEGER16 INTEGER32 INTEGER64
INTEGER128 INT8 INT16 INT32 INT64 INT128 STRING UNISTRING


(* Additional Builtins for ObjM2 *)

(* recognised by ObjM2 *)

(* rendered as builtins if recognised, as identifiers if not recognised *)

OBJECT NO YES


(* Additional Builtins for GNU Extensions to PIM and ISO *)

(* recognised by GM2 *)

(* rendered as builtins if recognised, as identifiers if not recognised *)

(* proper builtins *)

BITSET8 BITSET16 BITSET32 CARDINAL8 CARDINAL16 CARDINAL32 CARDINAL64 COMPLEX32
COMPLEX64 COMPLEX96 COMPLEX128 INTEGER8 INTEGER16 INTEGER32 INTEGER64 REAL8
REAL16 REAL32 REAL96 REAL128 THROW

(* pseudo-builtins *)

(* SYSTEM pseudo-module *)

byte


(* Additional Builtins for MOCKA Extensions to PIM *)

(* recognised by MOCKA *)

(* rendered as builtins if recognised, as identifiers if not recognised *)

(* proper builtins *)

LONGCARD SHORTCARD SHORTINT

(* pseudo-builtins *)

(* SYSTEM pseudo-module *)

byte


(* Additional Builtins for Aglet Extensions to ISO *)

(* recognised by Aglet *)

(* rendered as builtins if recognised, as identifiers if not recognised *)

BITSET8 BITSET16 BITSET32 CARDINAL8 CARDINAL16 CARDINAL32 INTEGER8 INTEGER16
INTEGER32


(* Additional Builtins for Gardens Point Extensions to ISO *)

(* recognised by Gardens Point *)
```

```
(* rendered as builtins if recognised, as identifiers if not recognised *)

(* proper builtins *)

ABORT SFLOAT SHORTREAL

(* pseudo-builtins *)

(* SYSTEM pseudo-module *)

BIN byte NEWPROCESS SAL


(* Additional Builtins for p1 Extensions to ISO *)

(* recognised by p1 *)

(* rendered as builtins if recognised, as identifiers if not recognised *)

(* pseudo-builtins *)

(* SYSTEM pseudo-module *)

bcd


(* Additional Builtins for Stony Brook Extensions to ISO *)

(* recognised by Stony Brook *)

(* rendered as builtins if recognised, as identifiers if not recognised *)

(* proper builtins *)

ACHAR BAND BNOT BOOL8 BOOL16 BOOL32 BOR BYTEBOOL CARDINAL8 CARDINAL16
CARDINAL32 CARDINAL64 DWORDBOOL INTEGER8 INTEGER16 INTEGER32 INTEGER64
LONGCARD NILPROC ROL ROR SHL SHORTCARD SHORTINT SHR UCHAR WORDBOOL

(* pseudo-builtins *)

CPUCOUNT EXITCODE BuildNumber DebuggerPresent OFFS UNREFERENCED_PARAMETER
SOURCEFILE SOURCELINE ASSERT ISASSERT EXCEPTADDR EXCEPT_INFO
SetUnhandledExceptionProc AttachDebugger AttachDebuggerOpt DoNotAttach
AttachExternal AttachAll OutputDebugMessage EnableCallTrace OutputCallTrace
TrapAccessViolations VA_START VA_ARG CLONE FIXME SWAPENDIAN BIGENDIAN
LITTLEENDIAN ATOMIC_CMPXCHG ATOMIC_XCHG ATOMIC_ADD MEMORY_FENCE


(* Additional Builtins for XDS Extensions to ISO *)

(* recognised by XDS *)

(* rendered as builtins if recognised, as identifiers if not recognised *)

(* proper builtins *)

ASH ASSERT DIFFADR_TYPE ENTIER INDEX LEN LONGCARD SHORTCARD SHORTINT

(* pseudo-builtins *)

(* SYSTEM pseudo-module *)

PROCESS NEWPROCESS BOOL8 BOOL16 BOOL32 CARD8 CARD16 CARD32 INT8 INT16 INT32
REF MOVE FILL GET PUT CC int unsigned size_t void

(* COMPILER pseudo-module *)

COMPILER OPTION EQUATION

(* end of file *)
```