```
-- file DIScanner.Mesa
-- last modified by
--                Sandman, May 5, 1978  8:17 AM
--                Barbara, July 10, 1978  2:55 PM

DIRECTORY
  AltoDefs: FROM "altodefs" USING [maxword],
  DIDefs: FROM "didefs",
  DILALRDefs: FROM "dilalrdefs" USING [
    endmarker, hashval, LALRTableHandle, Symbol, SymbolRecord, tokenCHAR,
    tokenDOT, tokenDOTS, tokenID, tokenLNUM, tokenNUM, tokenSTR,
    VocabHashEntry],
  DILitDefs: FROM "dilitdefs" USING [
    FindLiteral, FindLongLiteral, FindStringLiteral],
  IODefs: FROM "iodefs" USING [CR],
  StringDefs: FROM "stringdefs" USING [SubStringDescriptor];

DIScanner: PROGRAM
    IMPORTS DILitDefs
    EXPORTS DIDefs SHARES DILALRDefs =
  BEGIN
  OPEN DILALRDefs;

  InvalidCharacter: PUBLIC SIGNAL [index: CARDINAL] = CODE;

  InvalidNumber: PUBLIC SIGNAL [index: CARDINAL] = CODE;

  dHashTab: DESCRIPTOR FOR ARRAY OF DILALRDefs.VocabHashEntry;
  dScanTab: DESCRIPTOR FOR ARRAY CHARACTER [40C..177C] OF DILALRDefs.Symbol;
  vocab: STRING;
  dVocabIndex: DESCRIPTOR FOR ARRAY OF CARDINAL;

  NUL: CHARACTER = 0C;
  CR: CHARACTER = IODefs.CR;
  ControlZ: CHARACTER = 32C;                -- Bravo escape char

  text: STRING;        -- the input string

  desc: StringDefs.SubStringDescriptor;        -- initial buffer segment

  charIndex: CARDINAL;                -- index of current character
  currentChar: CHARACTER;        -- most recently scanned character

  Atom: PUBLIC PROCEDURE RETURNS [symbol: SymbolRecord] =
    BEGIN
    OPEN symbol;
    char, first, last: CHARACTER;
    uId: BOOLEAN;
    i, j, h: CARDINAL;
    s1, s2: CARDINAL;
    char ← currentChar;
    DO
      WHILE char IN [NUL..' ] DO
        SELECT char FROM
          ControlZ =>
            DO
              SELECT GetChar[] FROM
                NUL => GOTO EndFile;
                CR => EXIT;
                ENDCASE;
              ENDLOOP;
          NUL => GOTO EndFile;
          ENDCASE;
        char ← GetChar[];
        ENDLOOP;
      index ← charIndex;  value ← 0;
      SELECT char FROM
        'a, 'b, 'c, 'd, 'e, 'f, 'g, 'h, 'i, 'j, 'k, 'l, 'm,
        'n, 'o, 'p, 'q, 'r, 's, 't, 'u, 'v, 'w, 'x, 'y, 'z  =>
          BEGIN
          desc.offset ← index;
          i ← 0;
          DO
            char ← GetChar[];
            SELECT char FROM
              IN ['a..'z], IN ['A..'Z], IN ['0..'9] => i ← i+1;
```

```
        ENDCASE => EXIT;
      ENDLOOP;
    desc.length ← i+1;
    class ← tokenID;
    value ← DILitDefs.FindStringLiteral[@desc]; EXIT
    END;
  'A, 'B, 'C, 'D, 'E, 'F, 'G, 'H, 'I, 'J, 'K, 'L, 'M,
  'N, 'O, 'P, 'Q, 'R, 'S, 'T, 'U, 'V, 'W, 'X, 'Y, 'Z =>
    BEGIN
    desc.offset ← index;
    i ← 0; uId ← TRUE; first ← last ← char;
    DO
      char ← GetChar[];
      SELECT char FROM
        IN ['A..'Z] =>
          BEGIN last ← char;
          i ← i+1;
          END;
        IN ['a..'z], IN ['0..'9] =>
          BEGIN uId ← FALSE;
          i ← i+1;
          END;
        ENDCASE => EXIT;
      ENDLOOP;
    i ← i+1;
    IF uId THEN
      BEGIN
      h ← (LOOPHOLE[first, CARDINAL]*127 + LOOPHOLE[last, CARDINAL]) MOD hashval + 1;
      WHILE (j ← dHashTab[h].symptr) # 0 DO
        IF dVocabIndex[j]-(s2←dVocabIndex[j-1]) = i THEN
          FOR s1 IN [index .. index+i) DO
            IF text[s1] # vocab[s2] THEN EXIT;
            s2 ← s2+1;
            REPEAT
              FINISHED => GO TO reserved;
            ENDLOOP;
        IF (h ← dHashTab[h].link) = 0 THEN EXIT;
        ENDLOOP;
      END;
    desc.length ← i;
    class ← tokenID;
    value ← DILitDefs.FindStringLiteral[@desc]; EXIT
    EXITS
      reserved => BEGIN class ← j; EXIT END;
    END;
  '0, '1, '2, '3, '4, '5, '6, '7, '8, '9 =>
    BEGIN
    v, v10, v8: LONG INTEGER;
    scale: CARDINAL;
    valid, valid10, valid8, octal: BOOLEAN;
    vRep: ARRAY [0..SIZE[LONG INTEGER]) OF WORD; -- machine dependent
    v10 ← v8 ← 0; valid10 ← valid8 ← TRUE;
    WHILE char IN ['0..'9]
      DO
      IF valid10 THEN [v10, valid10] ← AppendDigit10[v10, char];
      IF valid8 THEN [v8, valid8] ← AppendDigit8[v8, char];
      char ← GetChar[];
      ENDLOOP;
    SELECT char FROM
      'B, 'C, 'b, 'c =>
        BEGIN
        class ← IF char = 'C OR char = 'c THEN tokenCHAR ELSE tokenNUM;
        v ← v8; valid ← valid8; octal ← TRUE;
        END;
      ENDCASE =>
        BEGIN
        class ← tokenNUM; v ← v10; valid ← valid10; octal ← FALSE;
        END;
    SELECT char FROM
      'B, 'C, 'D, 'b, 'c, 'd =>
        BEGIN
        char ← GetChar[];
        IF class = tokenNUM
          THEN
            BEGIN scale ← 0;
            WHILE char IN ['0..'9]
```

```
                DO
                scale ← 10*scale + CARDINAL[char-'0];
                char ← GetChar[];
                ENDLOOP;
              THROUGH [1 .. scale]  WHILE valid
                DO
                IF octal
                  THEN [v, valid] ← AppendDigit8[v, '0]
                  ELSE [v, valid] ← AppendDigit10[v, '0];
                ENDLOOP;
              END;
          END;
        ENDCASE;
      vRep ← LOOPHOLE[v];
      IF v <= AltoDefs.maxword
        THEN value ← DILitDefs.FindLiteral[vRep[0]]
        ELSE
          BEGIN
          IF class = tokenCHAR THEN valid ← FALSE;
          class ← tokenLNUM;
          value ← DILitDefs.FindLongLiteral[v];
          END;
      IF ~valid THEN SIGNAL InvalidNumber[index];
    EXIT
    END;

    '' =>
      BEGIN
      char ← GetChar[];
      class ← tokenCHAR;
      value ← DILitDefs.FindLiteral[LOOPHOLE[char, CARDINAL]];
      char ← GetChar[];  EXIT
      END;
    '" =>
      BEGIN
      desc.offset ← index+1;
      i ← 0;
      DO
        char ← GetChar[];
        IF char = NUL THEN char ← '";
        IF char = '" THEN
          BEGIN char ← GetChar[]; IF char # '" THEN EXIT; END;
        i ← i+1;
        ENDLOOP;
      desc.length ← i;
      class ← tokenSTR;
      value ← DILitDefs.FindStringLiteral[@desc];  EXIT
      END;
    '. =>
      BEGIN
      char ← GetChar[];
      IF char = '.
        THEN
          BEGIN  class ← tokenDOTS;
          char ← GetChar[];
          END
        ELSE class ← tokenDOT;
      EXIT
      END;
    ENDCASE =>
      BEGIN  class ← dScanTab[char];
      char ← GetChar[];
      IF class # 0 THEN EXIT ELSE SIGNAL InvalidCharacter[index];
      END;
  REPEAT
    EndFile =>  BEGIN  class ← endmarker;  value ← 0  END;
  ENDLOOP;
currentChar ← char;  RETURN
END;


-- Character source

GetChar: PROCEDURE RETURNS [CHARACTER] =
  BEGIN
  charIndex ← charIndex + 1;
```

```
    RETURN[IF charIndex >= text.length THEN NUL ELSE text[charIndex]];
    END;

  -- numerical conversion

  Digit: ARRAY CHARACTER ['0..'9] OF CARDINAL = [0,1,2,3,4,5,6,7,8,9];

  AppendDigit10: PROCEDURE [v: LONG INTEGER, digit: CHARACTER ['0..'9]]
      RETURNS [newV: LONG INTEGER, valid: BOOLEAN] =
    BEGIN
    MaxV: LONG INTEGER = 214748364;              -- (2**31-1)/10
    MaxD: CARDINAL = 7;                          -- (2**31-1) MOD 10
    d: [0..9] = Digit[digit];
    valid ← v < MaxV OR (v = MaxV AND d <= MaxD);
    newV ← 10*v + d;
    RETURN
    END;

  AppendDigit8: PROCEDURE [v: LONG INTEGER, digit: CHARACTER ['0..'9]]
      RETURNS [newV: LONG INTEGER, valid: BOOLEAN] =
    BEGIN
    MaxV: LONG INTEGER = 1777777777B;            -- (2**31-1)/8
    MaxD: CARDINAL = 7B;                         -- (2**31-1) MOD 8
    d: [0..9] = Digit[digit];
    valid ← (d < 8) AND (v < MaxV OR (v = MaxV AND d <= MaxD));
    newV ← 8*v + d;
    RETURN
    END;

  -- initialization/finalization

  ScanInit: PUBLIC PROCEDURE [string: STRING, tablePtr: DILALRDefs.LALRTableHandle] =
    BEGIN
      BEGIN  OPEN tablePtr.scantable;
      dHashTab ← DESCRIPTOR [hashtab];
      dScanTab ← DESCRIPTOR [scantab];
      vocab ← LOOPHOLE[@vocabbody, STRING];
      dVocabIndex ← DESCRIPTOR [vocabindex];
      END;
    desc.base ← text ← string;  desc.offset ← 0;
    charIndex ← 0;  currentChar ← text[0];
    RETURN
    END;

  ScanReset: PUBLIC PROCEDURE RETURNS [BOOLEAN] =
    BEGIN
    RETURN [charIndex >= text.length]
    END;


  END.
```