```
-- Double.Mesa  Edited by Sandman on May 12, 1978  2:14 PM

DIRECTORY
  DoubleDefs: FROM "doubledefs" USING [DAdd, DCompare, DSub],
  InlineDefs: FROM "inlinedefs" USING [
    BITSHIFT, LongCARDINAL, LongDiv, LongDivMod, LongMult],
  StringDefs: FROM "stringdefs" USING [AppendChar];

DEFINITIONS FROM InlineDefs, DoubleDefs;

Double: PROGRAM IMPORTS StringDefs EXPORTS DoubleDefs =
  PUBLIC BEGIN

  DDivide: PROCEDURE [num, den: LongCARDINAL]
    RETURNS [quotient, remainder: LongCARDINAL] =
    BEGIN
    qq: CARDINAL;
    count: [0..31);
    lTemp: LongCARDINAL;
    IF den.highbits = 0 THEN
      BEGIN
      [quotient.highbits, qq] ←
        LongDivMod[[lowbits:num.highbits,highbits:0],den.lowbits];
      [quotient.lowbits, remainder.lowbits] ←
        LongDivMod[[lowbits:num.lowbits,highbits:qq],den.lowbits];
      remainder.highbits ← 0;
      END
    ELSE
      BEGIN
      count ← 0;
      quotient.highbits ← 0;
      lTemp ← den;
      WHILE lTemp.highbits # 0 DO -- normalize
        lTemp.lowbits ←
          BITSHIFT[lTemp.lowbits,-1] + BITSHIFT[lTemp.highbits,15];
        lTemp.highbits ← BITSHIFT[lTemp.highbits,-1];
        count ← count + 1;
        ENDLOOP;
      qq ← LongDiv[num,lTemp.lowbits]; -- trial quotient
      qq ← BITSHIFT[qq,-count];
      lTemp ← LongMult[den.lowbits,qq]; -- multiply by trial quotient
      lTemp.highbits ← lTemp.highbits + den.highbits*qq;
      UNTIL DCompare[lTemp, num] # greater DO
        -- decrease quotient until product is small enough
        lTemp ← DSub[lTemp,den];
        qq ← qq - 1;
        ENDLOOP;
      quotient.lowbits ← qq;
      remainder ← DSub[num,lTemp];
      END;
    RETURN
    END;

  DMultiply: PROCEDURE [a,b: LongCARDINAL]
    RETURNS [product: LongCARDINAL] =
    BEGIN
    product ← LongMult[a.lowbits, b.lowbits];
    product.highbits ←
      product.highbits + a.lowbits*b.highbits + a.highbits*b.lowbits;
    RETURN
    END;

  DNeg: PROCEDURE [a: LongCARDINAL] RETURNS [LongCARDINAL] =
    BEGIN
    RETURN[DSub[[0,0],a]];
    END;

  DInc: PROCEDURE [a: LongCARDINAL] RETURNS [LongCARDINAL] =
    BEGIN
    RETURN[DAdd[[1,0],a]]
    END;

  AppendDouble: PROCEDURE [s: STRING, a: LongCARDINAL] =
    BEGIN  OPEN StringDefs;
    longZero: LongCARDINAL = [highbits:0, lowbits:0];
    long10: LongCARDINAL = [highbits:0, lowbits:10];
```

```
    xn: PROCEDURE =
      BEGIN
      charZero: CARDINAL = LOOPHOLE['0];
      r: LongCARDINAL;
      IF a # longZero THEN
        BEGIN
        [a, r] ← DDivide[a, long10];
        xn[];
        AppendChar[s, LOOPHOLE[r.lowbits+charZero, CHARACTER]];
        END;
      END;
    IF a = longZero THEN AppendChar[s, '0] ELSE xn[];
    RETURN
    END; --AppendDouble
END...
```