

-- Double.Mesa Edited by Sandman on August 11, 1977 2:59 PM

DIRECTORY

```
DoubleDefs: FROM "doubledefs",
InlineDefs: FROM "inlinedefs",
StringDefs: FROM "StringDefs";
```

DEFINITIONS FROM InlineDefs, DoubleDefs;

```
Double: PROGRAM IMPORTS StringDefs EXPORTS DoubleDefs =
PUBLIC BEGIN
```

```
DDivide: PROCEDURE [num, den: LongCARDINAL]
RETURNS [quotient, remainder: LongCARDINAL] =
BEGIN
qq, count: CARDINAL;
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[~um,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;
```

```
DAdd: PROCEDURE [a,b: LongCARDINAL] RETURNS [LongCARDINAL] =
BEGIN
t: CARDINAL = a.lowbits;
a.lowbits ← a.lowbits + b.lowbits;
a.highbits ← a.highbits + b.highbits;
IF a.lowbits < t THEN a.highbits ← a.highbits+1;
RETURN[a]
END;
```

```
DSub: PROCEDURE [a,b: longCARDINAL] RETURNS [LongCARDINAL] =
BEGIN
t: CARDINAL = a.lowbits;
a.lowbits ← a.lowbits - b.lowbits;
a.highbits ← a.highbits - b.highbits;
IF a.lowbits > t THEN a.highbits ← a.highbits-1;
RETURN[a]
END;
```

```
END;

DNeg: PROCEDURE [a: LongCARDINAL] RETURNS [LongCARDINAL] =
BEGIN
  IF (a.lowbits ← -a.lowbits) = 0 THEN a.highbits ← -a.highbits
  ELSE a.highbits ← BITNOT[a.highbits];
  RETURN[a];
END;

DInc: PROCEDURE [a: LongCARDINAL] RETURNS [LongCARDINAL] =
BEGIN
  IF (a.lowbits ← a.lowbits + 1) = 0 THEN
    a.highbits ← a.highbits + 1;
  RETURN[a]
END;

DCompare: PROCEDURE [a,b: LongCARDINAL] RETURNS [Comparison] =
BEGIN
  IF a = b THEN RETURN[equal];
  RETURN[SELECT a.highbits FROM
    < b.highbits => less,
    > b.highbits => greater,
  ENDCASE =>
    IF a.lowbits < b.lowbits THEN less ELSE greater]
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...
```