IBM SYSTEM/360
ASSEMBLY LANGUAGE INTERVAL ARITHMETIC SOFTWARE


APRIL 1972

GODDARD SPACE FLIGHT CENTER
GREENBELT, MARYLAND
IBM SYSTEM/360
ASSEMBLY LANGUAGE
INTERVAL ARITHMETIC SOFTWARE

E. J. Phillips
Computing and Software, Inc.
Data Systems Division

April 1972

Goddard Space Flight Center
Greenbelt, Maryland
# CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACKNOWLEDGEMENT</td>
<td>v</td>
</tr>
<tr>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>INTERVAL ARITHMETIC PACKAGE</td>
<td>2</td>
</tr>
<tr>
<td>II Language</td>
<td>2</td>
</tr>
<tr>
<td>III Purpose</td>
<td>2</td>
</tr>
<tr>
<td>III Method</td>
<td>2</td>
</tr>
<tr>
<td>IV Calling Sequence</td>
<td>2</td>
</tr>
<tr>
<td>V Mathematics Used in Package</td>
<td>7</td>
</tr>
<tr>
<td>VI Restrictions</td>
<td>9</td>
</tr>
<tr>
<td>VII Appendix (Sample problem and output)</td>
<td>10</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>16</td>
</tr>
</tbody>
</table>
ACKNOWLEDGMENT

The author wishes to gratefully acknowledge the assistance of Dr. Ronald Tost and other members of the Gesellschaft Für Mathemetik Und Datenverarbeitung, Bonn, Germany, who prepared and helped implement the software package on the IBM 360 system.
INTRODUCTION

This document describes some computer software designed to perform interval arithmetic. An interval is defined as the set of all real numbers between two given numbers including or excluding one or both endpoints. Interval arithmetic consists of the various elementary arithmetic operations defined on the set of all intervals, such as interval addition, subtraction, union, etc. The theoretical aspects of interval arithmetic are described in reference (2).

One of the main applications of interval arithmetic is in the area of error analysis of computer calculations. For example, it has been used successfully to compute bounds on sounding errors in the solution of linear algebraic systems (2), error bounds in numerical solutions of ordinary differential equations (1)(3) as well as integral equations and boundary value problems. The described software should enable users to implement algorithms of the type described in these references efficiently on the IBM 360 system.

The subroutines in this package were supplied to Goddard Space Flight Center by Dr. Ronald Tost, Institute für Numerische Datenverarbeitung under an exchange agreement with Dr. C. E. Velez, GSFC Code 552. Although the subroutines were coded in IBM 360 assembly language, they were designed so that calls may be made from FORTRAN programs. The sample problem in Section VII and all of the examples in Sections I to VI are given in FORTRAN to illustrate how this package may be used in FORTRAN programs. The subroutines were tested under varying conditions by Computing and Software, Inc., personnel.

The following pages explain in detail the arithmetic operations performed by this package.
INTERVAL ARITHMETIC PACKAGE

I. Language

Assembler Language Code for IBM/360.

II. Purpose

This package is designed to perform arithmetic operations using floating point (64 bits) interval arithmetic.

III. Method

The interval members are defined as COMPLEX * 16, by the user. The basic arithmetic operations are performed along with some basic set theory applications. The arithmetic functions are: addition, subtraction, multiplication, division, exponentiation. The absolute value of an interval and a negative interval can be created. Real and integer numbers can be converted into interval numbers. The intersection and union between two sets of interval numbers can be found. An illegal interval can be tested for. There are tests to determine if zero is contained in (e) an interval or if one interval is contained in another. The sum of two sets of interval numbers can be found by means of a scalar product. The above operations are accomplished by the following functions:

DLI, DRE, $IT$, $DT$, $ADD$, $SUB$, $MUL$, $DIV$, $POT$, $MIN$, $ABS$, $SCH$, $VER$, $IZER$, $ICOR$, $ENT$, $SUM$, $INP$.

IV. Calling Sequence

The interval numbers and functions are defined as COMPLEX * 16.

In the following set: A is an INTERVAL (AL, AR) B is an INTERVAL (BL, BR) C is an INTERVAL (CL, CR) D: REAL * 8 N: INTEGER * 4

Functions:

1. \( D = \text{DLI}(A) \quad D: = \text{AL} \quad \text{Left Corner of A} \)
   Ex. \( A = (2.5, 4.0) \)
   \( D = \text{DLI}(A) = 2.5 \)
2. **D = DRE(A) D: = AR** Right Corner of A  
   Ex. A = (2.5, 4.0)  
   D = DRE(A) = 4.0

3. **A = $IT$(N) A: = (N, N)** Produce an interval from an integer  
   Ex. N = 5  
   A = $IT$(N) = (5.0, 5.0)

4. **A = $DT$(D) A: = (D, D)** Produce an interval from a real number  
   Ex. D = 2.52  
   A = $DT$(D) = (2.52, 2.52)  
   A = $DT$(.5DO) = (0.5D0, 0.5D0)

5. **A = $ADD (B, C) A: = B + C** Addition  
   Ex. B = (3.0, 4.0)  
   C = (2.0, 2.0)  
   A = $ADD(B, C) = (3.0 + 2.0, 4.0 + 2.0) = (5.0, 6.0)

6. **A = $SUB (B, C) A: = B - C** Subtraction  
   Ex. B = (3.0, 4.0)  
   C = (2.0, 2.0)  
   A = $SUB(B, C) = (3.0 - 2.0, 4.0 - 2.0) = (1.0, 2.0)

7. **A = $MUL (B, C) A: = B * C** Multiplication  
   Ex. B = (3.0, 4.0)  
   C = (2.0, 2.5)  
   A = $MUL (B, C) = (MIN (6.0, 7.5, 8.0, 10.0), MAX (6.0, 7.5, 8.0, 10.0)) = (6.0, 10.0)

8. **A = $DIV (B, C) A: = B/C** Division  
   Ex. B = (3.0, 4.0) if 0 $\not\in$ (B, C)  
   C = (1.0, 2.0)  
   A = $DIV(B, C) = (3.0, 4.0) * (0.5, 1.0) = (MIN (1.5, 3.0, 2.0, 4.0), MAX (1.5, 3.0, 2.0, 4.0)) = (1.5, 4.0)
9. \( A = \$DOT (B, N) \quad A = B^N \) Exponentiation
   Ex. \( B = (3.0, 4.0) \)
   \( N = 2 \)
   \( A = \$DOT (B, N) = (9.0, 16.0) \)

10. \( A = \$MIN(B) \quad A = -B \) Negative Interval
    \( B = (3.0, 4.0) \)
    \( A = \$MIN(B) = (-4.0, -3.0) \)

11. \( A = \$ABS(B) \quad A = B \) if \( BL > 0 \) Absolute Value
    \( A = -B \) if \( BR < 0 \) else
    \( A = (0, \text{MAX}(BL, BR)) \)

Examples:

(1) \( B = (4.0, 4.25) \)
    \( A = \$ABS(B) = (4.0, 4.25) \)

(2) \( B = (-2.0, -1.0) \)
    \( A = \$ABS(B) = (1.0, 2.0) \)

(3) \( B = (-1.5, 4.0) \)
    \( A = \$ABS(B) = (0, \text{MAX}(-1.5, 4.0)) \)
    \( = (0, 4.0) \)

12. \( A = \$SCH(B, C) \quad A = (\text{MAX}((CL, BL), \text{MIN}(CR, BR)) \) Intersection
    Ex. \( B = (4.0, 5.0) \)
    \( \$SCH \) can produce illegal intervals. This can be
    \( C = (-1.5, 2.0) \)
    proved with ICOR.
    \( A = \$SCH(B, C) = (\text{MAX}(4.0, -1.5), \text{MIN}(5.0, 2.0)) \)
    \( = (4.0, 2.0) \) (- illegal interval)
    \( B = (1.0, 2.0) \)
    \( C = (-1.5, 5.0) \)
    \( A = \$SCH(B, C) = (1.0, 2.0) \)

13. \( A = \$VER(B, C) \quad A = (\text{MIN}(CL, BL), \text{MAX}(CR, BR)) \) Union
    \( B = (1.5, 2.5) \)
    \( C = (1.0, 5.0) \)
    \( A = \$VER(B, C) = (\text{MIN}(1.0, 1.5), \text{MAX}(5.0, 2.5)) \)
    \( = (1.0, 5.0) \)

14. IF (IZER (A)) \( ^\wedge, ^\wedge, ^\wedge \) Test to see if zero (0) is contained in A
    If IZER(A) \( < 0 \) GO TO\( ^\wedge \)
    If IZER(A) = 0 GO TO\( ^\wedge \)
    If IZER(A) > 0 GO TO\( ^\wedge \)
Test to see:

\[ \text{if } AR < 0 \quad (\text{if } AR < 0 , 0 \notin A) \]
\[ \text{if } AL > 0 \quad (\text{if } AL > 0 , 0 \notin A) \]
\[ \text{else } 0 \in A \quad \text{(zero contained in } A) \quad \text{null interval} \]

15. \text{IF (ICOR}(A)\text{)} \quad ^\wedge, ^\wedge\wedge, ^\wedge\wedge\wedge \quad \text{Interval Test}

\begin{align*}
\text{IF ICOR}(A) &< 0 \text{ GO TO } ^\wedge \\
\text{IF ICOR}(A) &= 0 \text{ GO TO } ^\wedge\wedge \\
\text{IF ICOR}(A) &> 0 \text{ GO TO } ^\wedge\wedge\wedge
\end{align*}

Test to see:

\[ \text{if } AL > AR \quad \text{(illegal interval)} \]
\[ \text{if } AL = AR \]
\[ \text{if } AL < AR \]

16. \$\text{ENT} – \text{Test on being contained in}

\begin{align*}
\text{CALL } $\text{ENTC} \ A, B, & \quad ^\wedge 1, & \quad ^\wedge 2, & \quad ^\wedge 3, & \quad ^\wedge 4 \\
^\wedge 1: & \ AL > BL \text{ and } AR < BR \quad \text{(A is contained in } B) \\
^\wedge 2: & \ AL = BL \text{ and } AR = BR \quad \text{(A and B are equal)} \\
^\wedge 3: & \ AL < BL \text{ and } AR > BR \quad \text{(B is contained in A)} \\
^\wedge 4: & \ A \text{ and } B \text{ are not fully contained in one another or are not contained in } \\
& \text{each other at all.}
\end{align*}

17. \$\text{SUM} – \text{Build a sum with a scalar product.}

\begin{align*}
\text{CALL } $\text{SUM}(A, I, J, K) \text{ where } I &= J, K \\
\$\text{SUM} \text{ store } A, I, J, K \text{ and sets up a loop made to perform addition.}
\end{align*}

Along with the call to \$\text{SUM} a multiplication statement has to follow immediately.

\text{Ex:}

\begin{align*}
\text{CALL } $\text{SUM} \ A, I, J, K \\
D &= $\text{MUL} \ B, C \\
B(1) &= (1.0, 2.0) \ C(1) = (2.0, 3.0) \\
B(2) &= (3.0, 4.0) \ C(2) = (4.0, 5.0) \\
B(3) &= (5.0, 6.0) \ C(3) = (6.0, 7.0)
\end{align*}

\text{Example:}

\begin{align*}
\text{CALL } $\text{SUM} \ A, I, 1, 3 \\
D &= $\text{MUL} \ B(I), C(I) \\
J &= 1, K = 3
\end{align*}

\text{Correspondent}

\begin{align*}
\text{AKKU} &= A \\
\text{DO 1 } I &= 1, 3
\end{align*}

\begin{align*}
D &= (0.0, 0.0) + (2.0, 6.0) \\
&= (2.0, 6.0) + (12.0, 20.0) \\
&= (14.0, 26.0) + (30.0, 42.0) \\
&= (44.0, 68.0)
\end{align*}
1. \[ \text{AKKU} = \text{AKKU} + \$\text{MUL} (\text{B(I)}, \text{C(I)}) \]
\[ \text{D} = \text{AKKU} \]

D, AKKU, and A are intervals; B, C are interval arrays. The sequence of \$\text{SUM} and \$\text{MUL} cannot contain interval functions.

18. \$\text{INP} \ A = \$\text{INP} ('\langleINPUT\rangle') \) conversion routine

\( \langle \text{INPUT} \rangle \) is an interval number, that is produced from an alphanumerical field of input values surrounded by quotation marks. The routine creates an interval from the set of numbers in the alphanumerical field by choosing the smallest number for the left endpoint and the largest number for the right endpoint. The interval number is greater than or equal to 1 \((\geq 1)\) in absolute value. The real or integer numbers are separated by commas but can contain no blanks. The beginning and end of the interval number must contain at least one blank. The left side of the output is the minimum of the real or integer numbers. The right side of the output contains the maximum of the real or integer numbers.

Example:

- \( \text{A} = \$\text{INP} ('1, 3, 1.56') \rightarrow \text{A} = (1.3, 1.56) \)
- \( \text{A} = \$\text{INP} ('-2., 4.') \rightarrow \text{A} = (-2., 4.) \)
- \( \text{A} = \$\text{INP} ('1') \rightarrow \text{No answer a blank doesn't preceed "1" on left side} \)
- \( \text{A} = \$\text{INP} ('1E45, 23, 2.5') \rightarrow \text{A} = (2.5, 1.E45) \)

The arithmetic functions can produce exponent overflow and divide check interrupts. Certain interval functions can produce illegal intervals and only two of the subroutines in this package can determine if the output is illegal, IZER and ICOR.

In all the arithmetic interval functions the input is tested to determine if it is illegal. No error messages are generated just a return to the calling program.

DEFINITION OF \( \langle \text{INPUT} \rangle \):

\( \langle \text{INPUT} \rangle : = ^* \langle \text{INTER} \rangle ^* \)
\( \langle \text{INTER} \rangle : = \langle \text{REAL} \rangle \)
\( : = \langle \text{INTER} \rangle, \langle \text{REAL} \rangle \)
\[ \langle \text{REAL} \rangle : = \langle \text{FLOAT} \rangle \]
\( : = \langle \text{FLOAT} \rangle \langle \text{EX} \rangle \langle \text{INTEG} \rangle \)
V. MATHEMATICS USED IN PACKAGE

If * is one of the symbols +, -, ·, /, the arithmetic operations of intervals are defined by:

\[ [a, b] \times [c, d] = \{ x \times y | a \leq x \leq b, c \leq y \leq d \} \]

\[ [a, b] + [c, d] = [a + c, b + d] \]

\[ [a, b] - [c, d] = [a - d, b - c] \]

\[ [a, b] \cdot [c, d] = [\min(ac, ad, bc, bd), \max(ac, ad, bc, bd)] \]

and if \[ 0 \notin [c, d] \] then

\[ [a, b] / [c, d] = [a, b] \cdot \begin{bmatrix} 1 & 1 \\ \frac{1}{d} & \frac{1}{c} \end{bmatrix} \]

Normalized floating point interval arithmetic is used in this package. Under IBM O/S 360 a floating point number is represented as

<table>
<thead>
<tr>
<th>bit</th>
<th>0</th>
<th>1</th>
<th>7</th>
<th>8</th>
<th>63</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>sign</td>
<td>characteristic exponent</td>
<td>fraction (binary) mantissa</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

where bit = 0 sign of number (0-positive, 1 negative)
\[ 1 - 7 = \text{the exponent indicating the power of 16 by which the fraction is multiplied. A zero in bit 1 indicates negative exponent, 1 indicates positive exponent. It is coded in excess 64-notation (decimal). That is, subtract decimal equivalent of characteristic by 64 to obtain the actual characteristic.} \]

\[ \text{e.g.} \quad 0 | 1000000 = 64 \]

\[ 64 - 64 = 0 = 16 \]

\[ 8 - 63 = \text{fraction or mantissa. The binary points of the fraction is just before bit position 8. That is to say that a 1 in bit position 8 represents } 2^{-1}, \text{ a 1 in bit position 9 represents } 2^{-2}, \text{ etc.} \]

The magnitudes of a floating point number is \( 16^{-65} \ (5.4 \times 10^{-79}) \)

\[ (1 - 16^{-14}) \times 16^{63} \ (7.2 \times 10^{75}) \]

Example:

\[ 1 \ 1000001 \ 01010100 \ 00000000 \ 00000000 \ 00000000 \text{ etc.} \]

<table>
<thead>
<tr>
<th>Sign</th>
<th>Characteristic</th>
</tr>
</thead>
<tbody>
<tr>
<td>(-1)</td>
<td>65</td>
</tr>
</tbody>
</table>

\[ 65 - 64 = 1 \]

\[ -1 \times 16^1 \times (2^{-2} + 2^{-4} + 2^{-6}) = -2^4 (2^{-2} + 2^{-4} + 2^{-6}) \]

\[ = - (2^2 + 2^0 + 2^{-2}) \]

\[ = - (4 + 1 + 0.25) = -5.25 \]

Normalized floating point numbers have a non-zero high order hexadecimal fraction digit. One or more high order hexadecimal fraction digits which are zero makes the number unnormalized. Normalization consists of shifting the fraction left until the high order hexadecimal digit is non-zero and reducing the characteristic by the number of hexadecimal digits shifted. For multiplication and division the operands are normalized before the operations are performed. For the remaining operands (+, -) the result is normalized.

\[ A = 436D1000 \ 00000000 = \text{normalized number} \]

\[ B = 4306D1000 \ 00000000 = \text{unnormalized} \]

\[ = 426D1000 \ 0000000 \text{ after normalization.} \]
VI. RESTRICTIONS

1. Only basic arithmetic operations and basic set theory applications are performed. No trigonometric operations are performed.

2. Uses long (64 bits) floating point interval arithmetic.

3. The arithmetic performed can produce illegal intervals.

4. The arithmetic performed can cause arithmetic overflow and divide check interrupts.
APPENDIX

LEVEL 20.1 (AUG 71)  OS/360 FORTRAN H  DATE 72.108/09.51.35

COMPILER OPTIONS - NAME= MAIN, OPT=01, LINECNT=58, SIZE=000K,
SOURCE= EBCDIC, NOLIST, NODECK, LOAD, MAP, NOEDIT, ID, XREF

<table>
<thead>
<tr>
<th>ISN</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>0002</td>
<td>DIMENSION C(20), D(20), E(20)</td>
</tr>
<tr>
<td>0003</td>
<td>WRITE(6,101)</td>
</tr>
<tr>
<td>0004</td>
<td>WRITE(6,101)</td>
</tr>
<tr>
<td>0005</td>
<td>I=1</td>
</tr>
<tr>
<td>0006</td>
<td>P=0.5D0</td>
</tr>
<tr>
<td>0007</td>
<td>A=(2.0D0, 5.0D0)</td>
</tr>
<tr>
<td>0008</td>
<td>B=(1.0D0, 8.0D0)</td>
</tr>
<tr>
<td>0009</td>
<td>N=3</td>
</tr>
<tr>
<td>0010</td>
<td>V=DLI(A)</td>
</tr>
<tr>
<td>0011</td>
<td>S=DRE(A)</td>
</tr>
<tr>
<td>0012</td>
<td>T=DLI(B)</td>
</tr>
<tr>
<td>0013</td>
<td>U=DRE(B)</td>
</tr>
<tr>
<td>0014</td>
<td>WRITE(6,203) V,S,T,U,N,P</td>
</tr>
<tr>
<td>0015</td>
<td>WRITE(6,222)</td>
</tr>
<tr>
<td>0016</td>
<td>D(1)=S$T$(N)</td>
</tr>
<tr>
<td>0017</td>
<td>WRITE(6,108)</td>
</tr>
<tr>
<td>0018</td>
<td>R=DLI(D(1))</td>
</tr>
<tr>
<td>0019</td>
<td>S=DRE(D(1))</td>
</tr>
<tr>
<td>0020</td>
<td>WRITE(6,202) I,R,S</td>
</tr>
<tr>
<td>0021</td>
<td>I=I+1</td>
</tr>
<tr>
<td>0022</td>
<td>D(2)=D$T$(P)</td>
</tr>
<tr>
<td>0023</td>
<td>WRITE(6,109)</td>
</tr>
<tr>
<td>0024</td>
<td>R=DLI(D(2))</td>
</tr>
<tr>
<td>0025</td>
<td>S=DRE(D(2))</td>
</tr>
<tr>
<td>0026</td>
<td>WRITE(6,202) I,R,S</td>
</tr>
<tr>
<td>0027</td>
<td>I=I+1</td>
</tr>
<tr>
<td>0028</td>
<td>D(3)=DADD(A,B)</td>
</tr>
<tr>
<td>0029</td>
<td>WRITE(6,102)</td>
</tr>
<tr>
<td>0030</td>
<td>R=DLI(D(3))</td>
</tr>
<tr>
<td>0031</td>
<td>S=DRE(D(3))</td>
</tr>
<tr>
<td>0032</td>
<td>WRITE(6,202) I,R,S</td>
</tr>
<tr>
<td>0033</td>
<td>I=I+1</td>
</tr>
<tr>
<td>0034</td>
<td>D(4)=D$T$(A,B)</td>
</tr>
<tr>
<td>0035</td>
<td>WRITE(6,103)</td>
</tr>
<tr>
<td>0036</td>
<td>R=DLI(D(4))</td>
</tr>
<tr>
<td>0037</td>
<td>S=DRE(D(4))</td>
</tr>
<tr>
<td>0038</td>
<td>WRITE(6,202) I,R,S</td>
</tr>
<tr>
<td>0039</td>
<td>I=I+1</td>
</tr>
<tr>
<td>0040</td>
<td>D(5)=D$T$(A,B)</td>
</tr>
<tr>
<td>0041</td>
<td>WRITE(6,104)</td>
</tr>
<tr>
<td>0042</td>
<td>R=DLI(D(5))</td>
</tr>
<tr>
<td>0043</td>
<td>S=DRE(D(5))</td>
</tr>
<tr>
<td>0044</td>
<td>WRITE(6,202) I,R,S</td>
</tr>
<tr>
<td>0045</td>
<td>I=I+1</td>
</tr>
<tr>
<td>0046</td>
<td>D(6)=D$T$(A,B)</td>
</tr>
<tr>
<td>0047</td>
<td>WRITE(6,105)</td>
</tr>
<tr>
<td>0048</td>
<td>R=DLI(D(6))</td>
</tr>
<tr>
<td>0049</td>
<td>S=DRE(D(6))</td>
</tr>
<tr>
<td>0050</td>
<td>WRITE(6,202) I,R,S</td>
</tr>
<tr>
<td>0051</td>
<td>WRITE(6,223)</td>
</tr>
<tr>
<td>0052</td>
<td>I=1</td>
</tr>
<tr>
<td>0053</td>
<td>C(1)=D$T$(A,N)</td>
</tr>
<tr>
<td>0054</td>
<td>WRITE(6,110)</td>
</tr>
<tr>
<td>0055</td>
<td>R=DLI(C(1))</td>
</tr>
</tbody>
</table>
ISN 0056  S=DREICII(I)
ISN 0057  WRITE(6,2002) I,R,S
ISN 0058      I=I+1
ISN 0059  C(2)=MIN(B,N)
ISN 0060  WRITE(6,111)
ISN 0061      R=DLCIC(1))
ISN 0062      S=DREIC(1))
ISN 0063  WRITE(6,2002) I,R,S
ISN 0064      I=I+1
ISN 0065  C(3)=MIN(A)
ISN 0066  WRITE(6,112)
ISN 0067      R=DLCI(C(1))
ISN 0068      S=DRE(C(1))
ISN 0069  WRITE(6,2002) I,R,S
ISN 0070      I=I+1
ISN 0071  C(4)=MIN(B)
ISN 0072  WRITE(6,113)
ISN 0073      R=DLCI(C(1))
ISN 0074      S=DRE(C(1))
ISN 0075  WRITE(6,2002) I,R,S
ISN 0076      I=I+1
ISN 0077  C(5)=ABS(A)
ISN 0078  WRITE(6,114)
ISN 0079      R=DLCI(C(1))
ISN 0080      S=DRE(C(1))
ISN 0081  WRITE(6,2002) I,R,S
ISN 0082      I=I+1
ISN 0083  C(6)=ABS(B)
ISN 0084  WRITE(6,115)
ISN 0085      R=DLCI(C(1))
ISN 0086      S=DRE(C(1))
ISN 0087  WRITE(6,2002) I,R,S
ISN 0088      I=I+1
ISN 0089  C(7)=SCH(A,B)
ISN 0090  WRITE(6,107)
ISN 0091      R=DLCI(C(1))
ISN 0092      S=DRE(C(1))
ISN 0093  WRITE(6,2002) I,R,S
ISN 0094      IF(ICO(C(1))) 10,20,20
ISN 0095  10 IF(R.GT.S) WRITE(6,190) R,S
ISN 0096      I=I+1
ISN 0097  20 C(8)=SCH(A,B)
ISN 0098  WRITE(6,106)
ISN 0099      R=DLCI(C(1))
ISN 0100      S=DRE(C(1))
ISN 0101  WRITE(6,2002) I,R,S
ISN 0102      IF(DER(1)) 28,30,29
ISN 0103  28 IF(S.LT.0) GO TO 35
ISN 0104  29 IF(R.GT.0) GO TO 35
ISN 0105  30 WRITE(6,210)
ISN 0106      C.TEST ON BEING_CONTAINED_IN
ISN 0107  35 WRITE(6,217)
ISN 0108  36 CALL SENT(A,0,636,637,638,639)
ISN 0109  37 WRITE(6,211)
ISN 0110      GO TO 40
ISN 0111  37 WRITE(6,212)
ISN 0114
   GO TO 40
ISN 0115
   WRITE(6,213)
ISN 0116
   GO TO 40
ISN 0117
   WRITE(6,214)
ISN 0118
   R=(0.000,0.000)
ISN 0119
   WRITE(6,101)
ISN 0120
   T=DL(R)
ISN 0121
   (=DRE(R)
ISN 0122
   WRITE(6,203) V,S,T,U,N,P
ISN 0123
   WRITE(6,217)
ISN 0124
   CALL SENT(A,B,641,642,643,644)
ISN 0125
   WRITE(6,211)
ISN 0126
   GO TO 44
ISN 0127
   WRITE(6,212)
ISN 0128
   GO TO 45
ISN 0129
   WRITE(6,213)
ISN 0130
   GO TO 45
ISN 0131
   WRITE(6,214)
ISN 0132
   GO TO 45
ISN 0133
   CALL $SUM(B,1,1,5)
ISN 0134
   F=$MUL(C(1),D(1))
ISN 0135
   R=DLIF)
ISN 0136
   S=DRE(F)
ISN 0137
   WRITE(6,218)
ISN 0138
   WRITE(6,215) R,S.
ISN 0139
   CONVERSION ROUTINE
ISN 0140
   C(1)=$INP( 1,3,1.56 )
ISN 0141
   C(2)=$INP( -2,4 )
ISN 0142
   C(3)=$INP(11 1)
ISN 0143
   C(4)=$INP( 1E45,23,2.5 1)
ISN 0144
   C(5)=$OTS(5,500)
ISN 0145
   WRITE(6,216)
ISN 0146
   DO 9 1=1,6
ISN 0147
   R=DLI(C(1))
ISN 0148
   S=DRE(C(1))
ISN 0149
   WRITE(6,2002) I,R,S
ISN 0150
   FORMAT/(X,15,'LEFT= ',D24.16,' RIGHT= ',D24.16)
ISN 0151
   101 FORMAT(' INTERVAL ARITHMETIC PACKAGE')
ISN 0152
   102 FORMAT('OA+B')
ISN 0153
   103 FORMAT('OA-A')
ISN 0154
   104 FORMAT('OA=R')
ISN 0155
   105 FORMAT('OA/B')
ISN 0156
   106 FORMAT('OA UNION B')
ISN 0157
   107 FORMAT('OA INTERSECT B')
ISN 0158
   108 FORMAT('OA B')
ISN 0159
   109 FORMAT('OTS(1)')
ISN 0160
   110 FORMAT('OA**N')
ISN 0161
   111 FORMAT('OPABS(A)')
ISN 0162
   112 FORMAT('0$MIN(R)')
ISN 0163
   113 FORMAT('0$ABS(R)')
ISN 0164
   114 FORMAT('0$MIN(R)')
ISN 0165
   115 FORMAT('0$ABS(R)')
ISN 0166
   190 FORMAT( ' THIS IS AN ILLEGAL INTERVAL (';,G24.16,' ',';,G24.16,' ')')
ISN 0167
   203 FORMAT( 'X1,';A= ',D24.16,' ',D24.16/1X,';B= ',D24.16,' ',D24.16/1X,';P= ',D24.16)
FORMAT(' ZERO IS CONTAINED IN THE FOLLOWING INTERVAL (',G24.16,')
FORMAT(' A IS CONTAINED IN B')
FORMAT(' A AND B ARE EQUAL')
FORMAT(' B IS CONTAINED IN A')
FORMAT(' A AND B ARE NOT FULLY CONTAINED IN EACH OTHER')
FORMAT(' LEFT= ',D24.16,2X,'RIGHT= ',D24.16)
FORMAT(' INPUT CONVERSION ROUTINE')
FORMAT(' QUESTION-TEST ON BEING CONTAINED IN')
FORMAT(' SUM PRODUCED WITH SCALAR PRODUCT')
FORMAT(' VALUES FOR ARRAY D(I)')
FORMAT(' VALUES FOR ARRAY C(I)')
STOP
END
### INTERVAL ARITHMETIC PACKAGE

<table>
<thead>
<tr>
<th>A</th>
<th>0.2000000000000000000000D 01</th>
<th>0.8000000000000000000000D 01</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>N</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>P</td>
<td>0.5000000000000000000000D 00</td>
<td></td>
</tr>
</tbody>
</table>

**SENT-TEST ON BEING CONTAINED IN**

A and B are not fully contained in each other.

**SUM PRODUCED WITH SCALAR PRODUCT**

| LEFT | -0.6850000000000000000000D 02 | RIGHT | 0.8730000000000000000000D 03 |

**$\text{INP CONVERSION ROUTINE}$**

| 1LEFT | 0.1300000000000000000000D 01 | 1RIGHT | 0.1560000000000000000000D 01 |
| 2LEFT | -0.2000000000000000000000D 01 | 2RIGHT | 0.4000000000000000000000D 01 |
| 3LEFT | -0.2000000000000000000000D 01 | 3RIGHT | 0.4000000000000000000000D 01 |
| 4LEFT | 0.2500000000000000000000D 01 | 4RIGHT | 0.1000000000000000000000D 46 |
| 5LEFT | 0.5500000000000000000000D 01 | 5RIGHT | 0.5500000000000000000000D 01 |
| 6LEFT | 0.9000000000000000000000D 01 | 6RIGHT | 0.9000000000000000000000D 01 |
## INTERVAL ARITHMETIC PACKAGE

### Values for Array D(i)

<table>
<thead>
<tr>
<th>Operator</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A + B$</td>
<td>0.13000000000000000 01</td>
</tr>
<tr>
<td>$A - B$</td>
<td>0.40000000000000000 01</td>
</tr>
<tr>
<td>$A \times B$</td>
<td>0.40000000000000000 02</td>
</tr>
<tr>
<td>$A / B$</td>
<td>0.50000000000000000 01</td>
</tr>
</tbody>
</table>

### Values for Array C(i)

<table>
<thead>
<tr>
<th>Operator</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A # N$</td>
<td>0.12500000000000000 03</td>
</tr>
<tr>
<td>$B # N$</td>
<td>0.51200000000000000 03</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\min(A)$</td>
<td>-0.20000000000000000 01</td>
</tr>
<tr>
<td>$\min(B)$</td>
<td>-0.10000000000000000 01</td>
</tr>
<tr>
<td>$\max(A)$</td>
<td>0.50000000000000000 01</td>
</tr>
<tr>
<td>$\max(B)$</td>
<td>0.80000000000000000 01</td>
</tr>
</tbody>
</table>

### Test on Being Contained in

- $A \subset B$: True
- $B \subset A$: False
References

