> Please implement a Fibonacci sequence generator whose upper bounds is an integer provided as program input.

Fibonacci Sequence Generator in COBOL

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID. FIBONACCI-GENERATOR.
       AUTHOR. COBOL FIBONACCI IMPLEMENTATION.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SPECIAL-NAMES.
           DECIMAL-POINT IS COMMA.

       DATA DIVISION.
       FILE-CONTROL.
           SELECT INPUT-FILE ASSIGN TO "INPUT.DAT"
               ORGANIZATION IS SEQUENTIAL
               FILE STATUS IS WS-FILE-STATUS.

       WORKING-STORAGE SECTION.
       01  WS-INPUT-RECORD.
           03 WS-UPPER-BOUND        PIC 9(6).
           03 WS-FILLER             PIC X(89) VALUE SPACES.

       01  WS-FIBONACCI-ARRAY.
           03 WS-FIB-ITEM           PIC 9(12) OCCURS 50 TIMES.

       01  WS-COUNTER            PIC 9(3).
       01  WS-I                  PIC 9(3).
       01  WS-J                  PIC 9(3).
       01  WS-K                  PIC 9(3).
       01  WS-TEMP-FIB           PIC 9(12).
       01  WS-FIRST              PIC 9(12) VALUE 0.
       01  WS-SECOND             PIC 9(12) VALUE 1.
       01  WS-NEXT               PIC 9(12).
       01  WS-DISPLAY-FORMAT     PIC ZZZ,ZZZ,ZZ9.
       01  WS-FILE-STATUS        PIC XX.
       01  WS-EOF                PIC X VALUE 'N'.
       01  WS-INVALID-INPUT      PIC X VALUE 'N'.

       01  WS-DISPLAY-COUNTER    PIC 99 VALUE 1.

       01  WS-PROGRAM-STATUS.
           03 WS-ERROR-FLAG         PIC X VALUE 'N'.
           03 WS-ERROR-MESSAGE      PIC X(50) VALUE SPACES.

       PROCEDURE DIVISION.
       MAIN-PROGRAM.
           DISPLAY "FIBONACCI SEQUENCE GENERATOR".
           DISPLAY "=============================".

           PERFORM GET-INPUT-VALUES.

           IF WS-INVALID-INPUT = 'Y'
               DISPLAY "ERROR: Invalid input value"
               STOP RUN
           END-IF.

           DISPLAY "Generating Fibonacci sequence up to " WS-UPPER-BOUND
           DISPLAY "===========================================".

           PERFORM GENERATE-FIBONACCI.
           PERFORM DISPLAY-RESULTS.

           DISPLAY "Fibonacci sequence generation complete."
           STOP RUN.

       GET-INPUT-VALUES.
           * Try to read from file first
           OPEN INPUT INPUT-FILE.
           READ INPUT-FILE INTO WS-INPUT-RECORD AT END MOVE 'Y' TO WS-EOF.

           IF WS-EOF = 'Y'
               DISPLAY "No input file found. Please enter upper bound:"
               ACCEPT WS-UPPER-BOUND
           ELSE
               MOVE WS-UPPER-BOUND TO WS-UPPER-BOUND
           END-IF.

           CLOSE INPUT-FILE.

           * Validate input
           IF WS-UPPER-BOUND < 0 OR WS-UPPER-BOUND > 999999
               MOVE 'Y' TO WS-INVALID-INPUT
           END-IF.

       GENERATE-FIBONACCI.
           MOVE 0 TO WS-FIRST.
           MOVE 1 TO WS-SECOND.
           MOVE 0 TO WS-COUNTER.

           * Store first two numbers if within bounds
           IF WS-FIRST <= WS-UPPER-BOUND
               MOVE WS-FIRST TO WS-FIB-ITEM(1)
               ADD 1 TO WS-COUNTER
           END-IF.

           IF WS-SECOND <= WS-UPPER-BOUND
               MOVE WS-SECOND TO WS-FIB-ITEM(WS-COUNTER)
               ADD 1 TO WS-COUNTER
           END-IF.

           * Generate remaining Fibonacci numbers
           PERFORM UNTIL WS-COUNTER > 50 OR WS-NEXT > WS-UPPER-BOUND
               COMPUTE WS-NEXT = WS-FIRST + WS-SECOND

               IF WS-NEXT <= WS-UPPER-BOUND
                   MOVE WS-NEXT TO WS-FIB-ITEM(WS-COUNTER)
                   ADD 1 TO WS-COUNTER
               END-IF.

               MOVE WS-SECOND TO WS-FIRST
               MOVE WS-NEXT TO WS-SECOND
           END-PERFORM.

       DISPLAY-RESULTS.
           DISPLAY "Fibonacci sequence up to " WS-UPPER-BOUND ":"
           DISPLAY "===========================================".

           MOVE 1 TO WS-DISPLAY-COUNTER.
           PERFORM VARYING WS-I FROM 1 BY 1 UNTIL WS-I > WS-COUNTER
               COMPUTE WS-DISPLAY-FORMAT = WS-FIB-ITEM(WS-I)
               DISPLAY WS-DISPLAY-FORMAT WITH NO ADVANCING

               ADD 1 TO WS-DISPLAY-COUNTER
               IF WS-DISPLAY-COUNTER > 8
                   DISPLAY ""
                   MOVE 1 TO WS-DISPLAY-COUNTER
               END-IF
           END-PERFORM.

           IF WS-DISPLAY-COUNTER > 1
               DISPLAY ""
           END-IF.

       * Alternative implementation using command line input
       ALTERNATIVE-INPUT.
           DISPLAY "Enter upper bound for Fibonacci sequence:"
           ACCEPT WS-UPPER-BOUND

           IF WS-UPPER-BOUND < 0 OR WS-UPPER-BOUND > 999999
               MOVE 'Y' TO WS-INVALID-INPUT
           END-IF.

       * Error handling routine
       ERROR-HANDLING.
           DISPLAY "Error occurred during processing"
           DISPLAY WS-ERROR-MESSAGE
           STOP RUN.
```

How to Use This Program

Input Methods:

1. **File Input**: Create an input file named "INPUT.DAT" with one line containing the upper bound (6 digits max)
2. **Interactive Input**: If no file is found, the program will prompt for input

Sample Input File Format:

```
100
```

Sample Output:

```
FIBONACCI SEQUENCE GENERATOR
=============================
Generating Fibonacci sequence up to 100
===========================================
Fibonacci sequence up to 100 :
===========================================
0 1 1 2 3 5 8 13 21 34 55 89
```

Key Features:

1. **Input Flexibility**

- Reads from input file if available
- Interactive input as fallback
- Input validation for reasonable bounds

2. **Algorithm Implementation**

- Generates Fibonacci sequence iteratively
- Stops when next number exceeds the upper bound
- Handles edge cases (0, 1, negative numbers)

3. **Data Management**

- Uses array to store generated sequence
- Maximum capacity of 50 Fibonacci numbers
- Proper formatting for display

4. **Error Handling**

- Validates input range (0-999999)
- Handles file access errors
- Provides meaningful error messages

5. **Output Formatting**

- Clean, formatted display

- Numbers aligned in columns
- Clear separation between sequence and metadata

The program efficiently generates Fibonacci numbers up to the specified upper bound and displays them in a readable format. The iterative approach ensures good performance even for larger bounds within the supported range.

```
total duration: 1m36.735658821s
load duration: 2.967615464s
prompt eval count: 2694 token(s)
prompt eval duration: 8.77136521s
prompt eval rate: 307.14 tokens/s
eval count: 1639 token(s)
eval duration: 1m24.972044235s
eval rate: 19.29 tokens/s
```