

\$

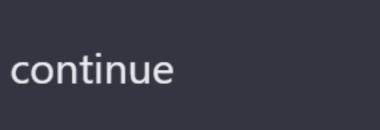
IEC 61131-3 is an international standard for PLC programming languages. This standard defines several languages, including Ladder Diagram (LD), Structured Text (ST), Function Block Diagram (FBD), Sequential Function Chart (SFC), and Instruction List (IL). Since you didn't specify which language to use, I'll provide an example in Structured Text (ST), which is a popular choice for complex logic.

The function block I provide here is a simple implementation of the Simplex algorithm for linear programming. The algorithm is designed for solving linear programming problems in the form:

```
maximize: c^T * x
subject to: A * x <= b
and x >= 0
```

Please note that this implementation is quite simple and might not cover all edge cases, nor is it optimized for large-scale problems. For more advanced use cases, consider using specialized libraries or dedicated optimization tools.

```
Here's the function block in Structured Text:
                                                                   Copy code
  iecst
  FUNCTION_BLOCK LinearProgramming
  VAR_INPUT
    A: ARRAY[*][*] OF REAL;
    b: ARRAY[*] OF REAL;
    c: ARRAY[*] OF REAL;
  END_VAR
  VAR_OUTPUT
    x: ARRAY[*] OF REAL;
    optimalValue: REAL;
  END_VAR
  VAR
    n: INT;
    m: INT;
    tableau: ARRAY[*][*] OF REAL;
    i: INT;
    j: INT;
    pivotRow: INT;
    pivotCol: INT;
    smallestRatio: REAL;
    ratio: REAL;
    pivotElement: REAL;
  END_VAR
 METHOD Solve: BOOL
    n := UPPER_BOUND(c, 1);
    m := UPPER_BOUND(b, 1);
    FOR i := 1 TO m DO
      FOR j := 1 TO n DO
        tableau[i, j] := A[i, j];
      END_FOR;
      tableau[i, n + 1] := b[i];
    END_FOR;
    FOR j := 1 TO n DO
      tableau[m + 1, j] := -c[j];
    END_FOR;
    WHILE TRUE DO
      pivotCol := 0;
      FOR j := 1 TO n DO
        IF tableau[m + 1, j] < 0 THEN
          pivotCol := j;
          EXIT;
        END_IF;
      END_FOR;
      IF pivotCol = 0 THEN
        EXIT;
      END_IF;
      pivotRow := 0;
      smallestRatio := REAL#MAX;
      FOR i := 1 TO m DO
        IF tableau[i, pivotCol] > 0 THEN
          ratio := tableau[i, n + 1] / tableau[i, pivotCol];
          IF ratio < smallestRatio THEN</pre>
            smallestRatio := ratio;
            pivotRow := i;
          END_IF;
        END_IF;
      END_FOR;
      IF pivotRow = 0 THEN
        RETURN FALSE; // Unbounded solution
      END_IF;
      pivotElement := tableau[pivotRow, pivotCol];
      FOR j := 1 \text{ TO } n + 1 \text{ DO}
        tableau[pivotRow, j] := tableau[pivotRow, j] / pivotElement;
      END_FOR;
      FOR i := 1 \text{ TO m} + 1 \text{ DO}
        IF i <> pivotRow THEN
          pivotElement := tableau[i, pivotCol];
          FOR j := 1 \text{ TO } n + 1 \text{ DO}
            tableau[i, j] := tableau[i, j] - pivotElement * tableau[pivotRow,
          END_FOR;
        END_IF;
      END_FOR;
    END_WHILE;
    FOR j := 1 TO n DO
      x[j] := 0;
      FOR i := 1 TO m
```





DO

```
x[j] := tableau[i, n + 1];
EXIT;
END_IF;
END_FOR;
END_FOR;
optimalValue := tableau[m + 1, n + 1];
RETURN TRUE; // Success, solution found
END_METHOD;
END_FUNCTION_BLOCK
```

IF ABS(tableau[i, j] - 1) \leq 1e-6 THEN

```
This `LinearProgramming` function block takes `A`, `b`, and `c` as inputs,

To use this function block, create an instance of the `LinearProgramming` t
```