1 POU: Day2

```
1   FUNCTION_BLOCK Day2 IMPLEMENTS IPuzzle
2   VAR CONSTANT
3       LINES               : DINT := 999;
4   END_VAR
5   VAR_OUTPUT
6       Finished            : BOOL := FALSE;
7       SolutionPart1       : UDINT;
8       SolutionPart2       : UDINT;
9   END_VAR
10  VAR
11      reader              : LineReader;
12      readingSuccess      : BOOL := FALSE;
13      reportNumbers       : ARRAY[0..LINES] OF STRING;
14  END_VAR
15
```

1.1 Method: IsLineValid

```
1   METHOD IsLineValid : BOOL
2   VAR_INPUT
3       buffer          : ARRAY[*] OF DINT;
4       bufferCount     : INT;
5   END_VAR
6   VAR_IN_OUT
7       invalidIndex    : DINT;
8   END_VAR
9   VAR
10      bufferStart     : INT := DINT_TO_INT(LOWER_BOUND(buffer, 1));
11      i               : INT;
12      direction       : INT := 0;
13      valid           : BOOL;
14  END_VAR
15
```

```
1   FOR i := bufferStart TO bufferCount - 2 DO
2       IF i = bufferStart THEN
3           direction := SIGN(buffer[i] - buffer[i + 1]);
4       END_IF
5
6       valid := IsStepValid(buffer[i], buffer[i + 1], direction);
7       IF valid = FALSE THEN
8           invalidIndex := i;
9           IsLineValid := FALSE;
10          RETURN;
11      END_IF
12  END_FOR
13  IsLineValid := TRUE;
14
```

1.2 Method: IsStepValid

```
METHOD IsStepValid : BOOL
VAR_INPUT
    numberA         : DINT;
    numberB         : DINT;
    direction       : DINT;
END_VAR
VAR
    difference      : DINT;
    stepSizeValid   : BOOL;
    directionValid  : BOOL;
END_VAR

```

```
difference := numberA - numberB;
stepSizeValid := ABS(difference) >= 1 AND ABS(difference) <= 3;
directionValid := direction = SIGN(difference);

IsStepValid := stepSizeValid AND directionValid;

```

1.3 Method: Solve

```
METHOD Solve
VAR CONSTANT
    BUFFER_LENGTH   : INT := 10;
END_VAR
VAR
    line            : STRING(255);
    lineIndex       : DINT;
    numberBuffer    : ARRAY[0..BUFFER_LENGTH] OF DINT;
    numberCount     : INT;
    valid           : BOOL;
    invalidStep     : DINT;
    fixed           : BOOL;
    tmpBuffer       : ARRAY[0..BUFFER_LENGTH] OF DINT;
    fixedReports    : DINT;
    tmp             : DINT;
END_VAR

```

```
reader(FilePath := 'inputs/day2.txt');
readingSuccess := reader.Done = TRUE AND reader.Error = FALSE;

// -*-* Day 2 *-*-
IF readingSuccess = TRUE AND Finished = FALSE THEN
    SolutionPart1 := 0;
    SolutionPart2 := 0;
    fixedReports := 0;

    // For each report
    FOR lineIndex := 0 TO LINES DO
        line := reader.ReadLines[lineIndex];
        numberCount := LineToNumbers(line := line, separator := ' ', numberBuffer := numberBuffer);
        valid := IsLineValid(numberBuffer, numberCount, invalidStep);

        // Count valid reports for solution 1
        IF valid = TRUE THEN
            SolutionPart1 := SolutionPart1 + 1;
            reportNumbers[lineIndex] := 'Valid';
        ELSE
```

```
21                  reportNumbers [ lineIndex ]  :=  CONCAT ( TO_STRING ( invalidStep ) ,  ': Invalid' ) ;
22              END_IF
23
24              // Try to fix invalid reports for solution 2
25              IF  valid  =  FALSE  THEN
26                  // Try to remove left value
27                  tmpBuffer  :=  numberBuffer ;
28                  ShiftArray ( ShiftDirection . LEFT ,  1 ,  invalidStep  +  1 ,  numberCount ,  tmpBuffer ) ;
29                  fixed  :=  IsLineValid ( tmpBuffer ,  numberCount  -  1 ,  tmp ) ;
30
31                  // Try to remove right value
32                  IF  fixed  =  FALSE  THEN
33                      tmpBuffer  :=  numberBuffer ;
34                      ShiftArray ( ShiftDirection . LEFT ,  1 ,  invalidStep  +  2 ,  numberCount ,  tmpBuffer )
    ;
35                      fixed  :=  IsLineValid ( tmpBuffer ,  numberCount  -  1 ,  tmp ) ;
36                  END_IF
37
38                  IF  fixed  =  TRUE  THEN
39                      fixedReports  :=  fixedReports  +  1 ;
40                      reportNumbers [ lineIndex ]  :=  CONCAT ( TO_STRING ( invalidStep ) ,  ': Fixed' ) ;
41                  END_IF
42              END_IF
43          END_FOR
44          SolutionPart2  :=  SolutionPart1  +  DINT_TO_UDINT ( fixedReports ) ;
45          Finished  :=  TRUE ;
46      END_IF
47
```