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       bufferLength    : 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      direction       : INT := 0;
12      valid           : BOOL;
13      i               : INT;
14      current         : DINT;
15      next            : DINT;
16  END_VAR
17
```

```
1   FOR i := bufferStart TO bufferLength - 2 DO
2       current := buffer[i];
3       next := buffer[i + 1];
4
5       IF i = bufferStart THEN
6           direction := SIGN(current - next);
7       END_IF
8
9       valid := IsStepValid(current, next, direction);
10      IF valid = FALSE THEN
11          invalidIndex := i;
12          IsLineValid := FALSE;
13          RETURN;
14      END_IF
15  END_FOR
16
17  IsLineValid := TRUE;
18
```

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: Reset

```
METHOD Reset
VAR
    i : DINT ;
END_VAR

```

```
Finished := FALSE ;
SolutionPart1 := 0 ;
SolutionPart2 := 0 ;

FOR i := 0 TO LINES DO
    reportNumbers [ i ] := '' ;
END_FOR

```

1.4 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 ;
    i               : INT ;
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);

            // Solution 1: Count valid reports
            IF valid = TRUE THEN
                SolutionPart1 := SolutionPart1 + 1;
                reportNumbers[lineIndex] := 'Valid';
            ELSE
                reportNumbers[lineIndex] := CONCAT(TO_STRING(invalidStep), ': Invalid');
            END_IF

            // Solution 2: Try to fix invalid reports
            IF valid = FALSE THEN
                // Try remove the left (+1) or right (+2) value of the invalid step by shifitng the array
                FOR i := 1 TO 2 DO
                    tmpBuffer := numberBuffer;
                    ShiftArray(ShiftDirection.LEFT, 1, invalidStep + i, numberCount, tmpBuffer);
                    fixed := IsLineValid(tmpBuffer, numberCount - 1, tmp);

                    IF fixed = TRUE THEN
                        fixedReports := fixedReports + 1;
                        reportNumbers[lineIndex] := CONCAT(TO_STRING(invalidStep), ': Fixed');
                        EXIT;
                    END_IF
                END_FOR
            END_IF
        END_FOR
        SolutionPart2 := SolutionPart1 + DINT_TO_UDINT(fixedReports);
        finished := TRUE;
    END_IF

```

1.5 Interface property: IsFinished

```
    PROPERTY IsFinished : BOOL

```

1.5.1 'get' accessor: Get

```
    IsFinished := finished;

```