EXAMPLE PROGRAM 2D GEOGRAPHIC DISTANCE REPORT

1. Requirements

Design and implement an Ada program that determines the geographical distance between two points located on a geographic plane. The points are referenced using the standard {X,Y} Cartesian coordinate system founded on a 0 origin (see Figure 1). The geographic space of interest measures 100x100. X and Y Coordinates can therefore range from 0.0 to 100.0 inclusive. Distances should be determined to within 4 decimal places.

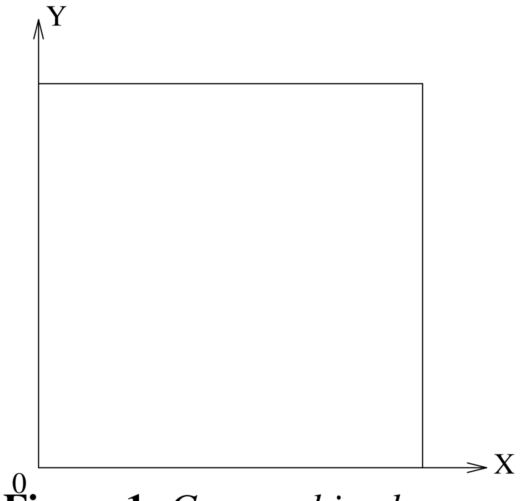

Figure 1: *Geographic plane*

2. Design

The distance between two two-dimensional points is given by:

$$Distance = \sqrt{x^2+y^2}$$

where x and y respectively represent the differences between the identifying pairs of X and Y coordinates for the two points. We can identify a number of operations here as indicated by the top-down analysis presented in Table 1.

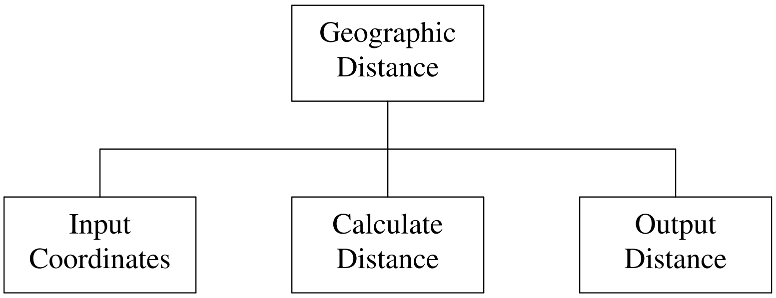

Table 1: *Top-down analysis*

These operations are simple enough to be combined into a single procedure:

1) `GEO_DIST_2D` (top level procedure) - input data, and calculate and output distance.

Name	Description	Type	Range
X1	Input variable	FLOAT	0.0..100.0
Y1	Input variable	FLOAT	0.0..100.0
X2	Input variable	FLOAT	0.0..100.0
Y2	Input variable	FLOAT	0.0..100.0
X_DIFF	Global variable	FLOAT	Default
Y_DIFF	Global variable	FLOAT	Default
DISTANCE	Global variable	FLOAT	Default

The design of this procedure is indicated by the Nassi-Shneiderman chart presented in Table 2.

Input X1 and Y1 for point 1
Input X2 and Y2 for point 2
Calculate difference X1 and X2 (X_DIFF)
Calculate difference Y1 and Y2 (Y_DIFF)
Calculate DISTANCE (SQRT(X_DIFF^2 + Y_DIFF^2))
Output DISTANCE

Table 2: *Nassi-Shneiderman Chart*

3. Implementation

Implementation as shown in Table 3:

```
-- 2D GEOGRAPHIC DISTANCE
-- 4 August 1997
-- Frans Coenen
-- Dept Computer Science, University of Liverpool

with TEXT_IO, GENERIC_ELEMENTARY_FUNCTIONS;
use TEXT_IO;

procedure GEO_DIST_2D is
        X1, Y1, X2, Y2: FLOAT range 0.0..100.0;
        X_DIFF, Y_DIFF, DISTANCE: FLOAT;
        package FLOAT_INOUT is new FLOAT_IO(FLOAT);
        use FLOAT_INOUT;
        package MATH_FUNC is new GENERIC_ELEMENTARY_FUNCTIONS(FLOAT);
        use MATH_FUNC;
```

Table 3: *Implementation*

```
begin
-- Input coordinates for point 1
        PUT_LINE("Input X and Y coordinates for point 1: ");
        GET(X1);
        GET(Y1);
-- Input coordinates for point 2
        PUT_LINE("Input X and Y coordinates for point 2: ");
        GET(X2);
        GET(Y2);
-- Calculate differences and distance
        X_DIFF := abs(X1-X2);
        Y_DUFF := abs(Y1-Y2);
        DISTANCE := SQRT(X_DIFF**2 + Y_DIFF**2);
-- Output distance
        PUT("Distance is: ");
        PUT(DISTANCE, FORE=>3, AFT=>4, EXP=>0);
        NEW_LINE;
end GEO_DIST_2D;
```

Table 3 continued

4. Testing

BVA testing: to test limits of input values. Appropriate values for `X1`, `Y1`, `X2` and `Y2` are -0.1, 0.1, 99,9 and 100.1 (just above and below the range limits). Test cases of this form are presented in the Table 4 where the * symbol indicates where no further input is required as the code is expected to fail after the previous input. (Results presented in Table 5.)

TEST CASE				EXPECTED RESULT
X1	Y1	X2	Y2	DISTANCE
-0.1	*	*	*	CONSTRAINT_ERROR
100.1	*	*	*	CONSTRAINT_ERROR
0.1	-0.1	*	*	CONSTRAINT_ERROR
99.9	100.1	*	*	CONSTRAINT_ERROR
0.1	0.1	-0.1	*	CONSTRAINT_ERROR
99.9	99.9	100.1	*	CONSTRAINT_ERROR
0.1	0.1	0.1	-0.1	CONSTRAINT_ERROR
99.9	99.9	99.9	100.1	CONSTRAINT_ERROR
0.1	99.9	99.9	0.1	141.1385
99.9	0.1	0.1	99.9	141.1385

Table 4: *BVA test cases*

```
kuban-209 $ geo_dist_2d
Input X and Y coordinates for point 1:
-0.1
Ada-runtime: Exception CONSTRAINT_ERROR propagated out of main.

kuban-210 $ geo_dist_2d
Input X and Y coordinates for point 1:
100.1
Ada-runtime: Exception CONSTRAINT_ERROR propagated out of main.

kuban-211 $ geo_dist_2d
Input X and Y coordinates for point 1:
0.1 -0.1
Ada-runtime: Exception CONSTRAINT_ERROR propagated out of main.

kuban-212 $ geo_dist_2d
Input X and Y coordinates for point 1:
99.9 100.1
Ada-runtime: Exception CONSTRAINT_ERROR propagated out of main.

kuban-213 $ geo_dist_2d
Input X and Y coordinates for point 1:
0.1 0.1
Input X and Y coordinates for point 2:
-0.1
Ada-runtime: Exception CONSTRAINT_ERROR propagated out of main.

kuban-214 $ geo_dist_2d
Input X and Y coordinates for point 1:
99.9 99.9
Input X and Y coordinates for point 2:
100.1
Ada-runtime: Exception CONSTRAINT_ERROR propagated out of main.

kuban-215 $ geo_dist_2d
Input X and Y coordinates for point 1:
0.1 0.1
Input X and Y coordinates for point 2:
0.1 -0.1
Ada-runtime: Exception CONSTRAINT_ERROR propagated out of main.

kuban-216 $ geo_dist_2d
Input X and Y coordinates for point 1:
99.9 99.9
Input X and Y coordinates for point 2:
99.9 100.1
Ada-runtime: Exception CONSTRAINT_ERROR propagated out of main.
```

Table 5: *BVA test output*

```
kuban-222 $ geo_dist_2d
Input X and Y coordinates for point 1:
0.1 99.9
Input X and Y coordinates for point 2:
99.9 0.1
Distance is: 141.1385

kuban-223 $ geo_dist_2d
Input X and Y coordinates for point 1:
99.9 0.1
Input X and Y coordinates for point 2:
0.1 99.9
Distance is: 141.1385
```

Table 5 continued

Arithmetic testing: Code includes three arithmetic expressions, two difference calculations and the Pythagoras calculation. Commencing with the two difference calculations, we note that each is carried out independently from the other and thus can be tested simultaneously. Each should be tested with combinations of positive and zero sample values.

Turning to the Pythagoras calculation this should be tested with sample values for `X_DIFF` and `Y_DIFF` comprising all possible combinations zero and positive value (we cannot have a negative difference) Test cases required to check the difference calculation already identify the situation where both difference values are zero and both difference values are positive. We therefore simply need to derive two test cases where one or the other difference value is zero .

An appropriate set of test cases is given in Table 6 (results presented in table 7).

TEST CASE				EXPECTED RESULT
X1	Y1	X2	Y2	DISTANCE
0.0	0.0	0.0	0.0	0.0000
0.0	0.0	10.0	10.0	14.1421
10.0	10.0	0.0	0.0	14.1421
10.0	10.0	10.0	10.0	0.0000
20.0	10.0	20.0	30.0	20.0000
30.0	20.0	10.0	20.0	20.0000

Table 6: *Arithmetic test cases*

```
kuban-226 $ geo_dist_2d
Input X and Y coordinates for point 1:
0.0 0.0
Input X and Y coordinates for point 2:
0.0 0.0
Distance is:   0.0000

kuban-227 $ geo_dist_2d
Input X and Y coordinates for point 1:
0.0 0.0
Input X and Y coordinates for point 2:
10.0 10.0
Distance is:  14.1421

kuban-228 $ geo_dist_2d
Input X and Y coordinates for point 1:
10.0 10.0
Input X and Y coordinates for point 2:
0.0 0.0
Distance is:  14.1421

kuban-229 $ geo_dist_2d
Input X and Y coordinates for point 1:
10.0 10.0
Input X and Y coordinates for point 2:
10.0 10.0
Distance is:   0.0000

kuban-230 $ geo_dist_2d
Input X and Y coordinates for point 1:
20.0 10.0
Input X and Y coordinates for point 2:
20.0 30.0
Distance is:  20.0000

kuban-231 $ geo_dist_2d
Input X and Y coordinates for point 1:
30.0 20.0
Input X and Y coordinates for point 2:
10.0 20.0
Distance is:  20.0000
```

Table 7: *Arithmetic test case output*

Limit testing|fR: We should also test the operations of the systems at the limits for the input data. Notionally there are four tests for each difference calculation covering every combination of 0.0 and 100.0 input. However, the above arithmetic test cases (Table 7) have considered the case where inputs are 0.0. Therefore we only need to test combinations where one or both of the input is 100.0. An appropriate set of test cases is given in table 8. (Results in Table 9.)

TEST CASE				EXPECTED RESULT
X1	Y1	X2	Y2	DISTANCE
100.0	0.0	0.0	100.0	141.4214
0.0	100.0	100.0	0.0	141.4214
100.0	100.0	100.0	100.0	0.0000

Table 9: *Limit test cases*

```
kuban-379 $ geo_dist_2d
Input X and Y coordinates for point 1:
100.0 0.0
Input X and Y coordinates for point 2:
0.0 100.0
Distance is: 141.4214

kuban-379 $ geo_dist_2d
Input X and Y coordinates for point 1:
0.0 100.0
Input X and Y coordinates for point 2:
100.0 0.0
Distance is: 141.4214

kuban-379 $ geo_dist_2d
Input X and Y coordinates for point 1:
100.0 100.0
Input X and Y coordinates for point 2:
100.0 100.0
Distance is:   0.0000
```

Table 9: *Limit test case output*

Fata validation testing:: Finally we should carry out some data validation using spurious inputs (see Table 10).

```
Input X and Y coordinates for point 1:
10.0 20.0 30.0
Input X and Y coordinates for point 2:
40.0 50.0
Distance is:  28.2843
kuban-382 $ 50.0
ksh: 50.0:  not found

kuban-383 $ geo_dist_2d
Input X and Y coordinates for point 1:
10.0 2e.0

Ada-runtime: Exception FLOAT_INPUT_ERROR_PERIOD_NOT_READ
propagated out of main.
kuban-384 $ .0
ksh: .0:  not found
```

Table 10: *Data validation test case output*