EXAMPLE PROGRAM PYTHAGORAS REPORT

1. Requirements

Given the length of the "opposite" (O) and "adjacent" (A) sides of a right angled triangle (Figure 1) determine the "hypotenuse" (H) using Pythagoras theorem:

$$H^2 = O^2 + A^2$$

Assume sides are given as floating point numbers between 0.0 and 100.0 and that an accuracy of 4 decimal places is required.

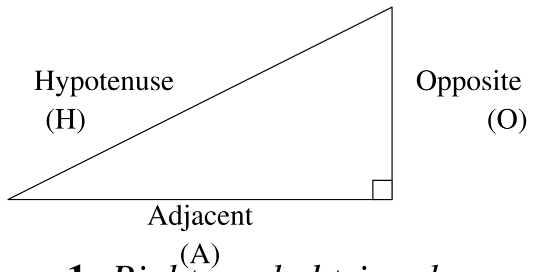

Figure 1: *Right-angled triangle*

2. Design

A top down analysis is presented in Table 1.

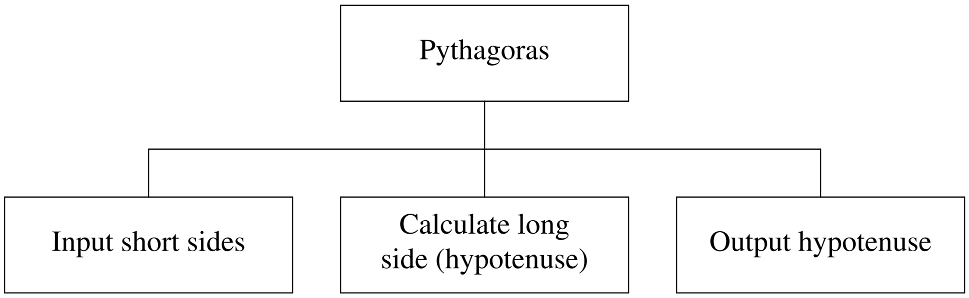

Table 1: *Top down analysis*

These tasks are trivial enough to be encompassed by a single procedure:

1) `PYTHAGORAS` (top level procedure) - read in short sides and calculate and output long side.

Name	Description	Type	Range
OPPOSITE	Input variable	FLOAT	0.0..100.0
ADJACENT	Input variable	FLOAT	0.0..100.0
HYPOTENUSE	Global variable	FLOAT	0.0..100.0

A Nassi-Shneiderman design is presented in Table 2.

Input OPPOSITE side
Input ADJACENT side
Calculate HYPOTENUSE
Output HYPOTENUSE

Table 1: *Nassi-Shneiderman Chart*

3. Implementation

Implementation is as shown in Table 3:

```
-- PYTHAGORAS
-- 6 August 1997
-- Frans Coenen
-- Dept Computer Science, University of Liverpool

with TEXT_IO, GENERIC_ELEMENTARY_FUNCTIONS;
use TEXT_IO;

procedure PYTHAGORAS is
        OPPOSITE, ADJACENT: FLOAT range 0.0..100.0;
        HYPOTENUSE: FLOAT;
        package FLOAT_INOUT is new FLOAT_IO(FLOAT);
        use FLOAT_INOUT;
        package MATH_FUNC is new GENERIC_ELEMENTARY_FUNCTIONS(FLOAT);
        use MATH_FUNC;
begin
-- Input opposite and adjacent side
        PUT_LINE("Input opposite and adjacent sides: ");
        GET(OPPOSITE);
        GET(ADJACENT);
-- Calculate hypotenuse
        HYPOTENUSE := OPPOSITE**2 + ADJACENT**2;
        HYPOTENUSE := SQRT(HYPOTENUSE);
-- Output hypotenuse
        PUT("Hypotenuse is: ");
        PUT(HYPOTENUSE, FORE=>3, AFT=>4, EXP=>0);
        NEW_LINE;
end PYTHAGORAS;
```

Table 3: *Implementation*

4. Testing

BVA Testing: Use Boundary Value Analysis (BVA) testing to anlyse limits of input variables. Suitable boundary values for both `O` and `A` are -0.1, 0.1, 99.9, 100.1 (just above and below the range limits). An appropriate set of test cases is given in the Table 4, where the * symbol indicates that no further input is required as we expect the code to fail after the previous input. Note that values for `ADJACENT` can only be tested given valid values for `OPPOSITE`.

TEST CASE		EXPECTED RESULT
OPPOSITE	ADJACENT	HYPOTENUSE
-0.1	*	CONSTRAINT_ERROR
0.1	-0.1	CONSTRAINT_ERROR
0.1	0.1	0.1414
99.9	99.9	141.2799
99.9	100.1	CONSTRAINT_ERROR
101.1	*	CONSTRAINT_ERROR

Table 4: *BVA test cases*

The output produced when the above cases are run is given in Table 5.

```
kuban-186 $ pythagoras
Input opposite and adjacent sides:
-0.1
Ada-runtime: Exception CONSTRAINT_ERROR raised in
/tmp_mnt/home/staff2/ra/frans/public_html/2CS21/week4/pythagoras.ada
on line 15. Floating rangecheck error: 0 not in range 0..0.

kuban-187 $ pythagoras
Input opposite and adjacent sides:
0.1
-0.1
Ada-runtime: Exception CONSTRAINT_ERROR raised in
/tmp_mnt/home/staff2/ra/frans/public_html/2CS21/week4/pythagoras.ada
on line 16. Floating rangecheck error: 0 not in range 0..0.

kuban-188 $ pythagoras
Input opposite and adjacent sides:
0.1
0.1
Hypotenuse is:   0.1414

kuban-195 $ pythagoras
Input opposite and adjacent sides:
99.9
99.9
Hypotenuse is: 141.2799
```

Table 5: *BVA test case output*

```
kuban-193 $ pythagoras
Input opposite and adjacent sides:
99.9
100.1
Ada-runtime: Exception CONSTRAINT_ERROR raised in
/tmp_mnt/home/staff2/ra/frans/public_html/2CS21/week4/pythagoras.ada
on line 16. Floating rangecheck error: 0 not in range 0..0.

kuban-194 $ pythagoras
Input opposite and adjacent sides:
101.1
Ada-runtime: Exception CONSTRAINT_ERROR raised in
/tmp_mnt/home/staff2/ra/frans/public_html/2CS21/week4/pythagoras.ada
on line 15. Floating rangecheck error: 0 not in range 0..0.
```

Table 5 continued

Limit testing: Test limits of input values using all four possible combinations using a set of test cases of the form presented in Table 6 (results presented in Table 7).

TEST CASE		EXPECTED RESULT
OPPOSITE	ADJACENT	HYPOTENUSE
0.0	0.0	0.0
0.0	100.0	100.0
100.0	0.0	100.0
100.0	100.0	141.4214

Table 6: *Limit test cases*

```
kuban-329 $ pythagoras
Input opposite and adjacent sides:
0.0 0.0
Hypotenuse is:   0.0000

kuban-330 $ pythagoras
Input opposite and adjacent sides:
0.0 100.0
Hypotenuse is: 100.0000

kuban-331 $ pythagoras
Input opposite and adjacent sides:
100.0 0.0
Hypotenuse is: 100.0000

kuban-332 $ pythagoras
Input opposite and adjacent sides:
100.0 100.0
Hypotenuse is: 141.4214
```

Table 7: *Limit test case output*

Arithmetic testing: Use straightforward numeric testing to test calculation. Normally this would require positive, zero and negative sample values for each input. However, in this case we need only to consider positive and zero values. The set of test cases given in Table 8 would be appropriate (results in table 9).

TEST CASE		EXPECTED RESULT
OPPOSITE	ADJACENT	HYPOTENUSE
0.0	0.0	0.0
0.0	10.0	10.0
10.0	0.0	10.0
10.0	10.0	

Table 8: *Arithmetic test cases*

```
kuban-310 $ pythagoras
Input opposite and adjacent sides:
0.0
0.0
Hypotenuse is:   0.0000

kuban-313 $ pythagoras
Input opposite and adjacent sides:
0.0
10.0
Hypotenuse is:  10.0000

kuban-314 $ pythagoras
Input opposite and adjacent sides:
10.0
0.0
Hypotenuse is:  10.0000

kuban-315 $ pythagoras
Input opposite and adjacent sides:
10.0 10.0
Hypotenuse is:  14.1421
```

Table 9: *Arithmetic test case output*

Data validation testing: Results of a number of data validation tests are presented in Table 10.

```
kuban-319 $ pythagoras
Input opposite and adjacent sides:
a
Ada-runtime: Exception FLOAT_INPUT_ERROR_DIGIT_NOT_READ
propagated out of main.

kuban-320 $ pythagoras
Input opposite and adjacent sides:
10.0 1e.0
Ada-runtime: Exception FLOAT_INPUT_ERROR_PERIOD_NOT_READ
propagated out of main.
kuban-321 $ .0
ksh: .0:  not found

kuban-323 $ pythagoras
Input opposite and adjacent sides:
10.0 20.0 30.0
Hypotenuse is:   4.7287
kuban-324 $ 30.0
ksh: 30.0:  not found
```

Table 10: *Data validation test case output*