EXAMPLE PROGRAM METRES TO YARDS FEET AND INCHES CONVERSION

1. Requirements

Design and create an Ada program that, when presented with a distance given in Metres converts it to a distance measure comprising Yards, Feet and Inches Assume 1 Metre = 39.37 Inches). Output the result in whole Yards, Feet and inches. Assume that the maximum distance to be processed is 1x10^6 Metres.

2. Design

A top down analysis of the problem is given in Table 1.

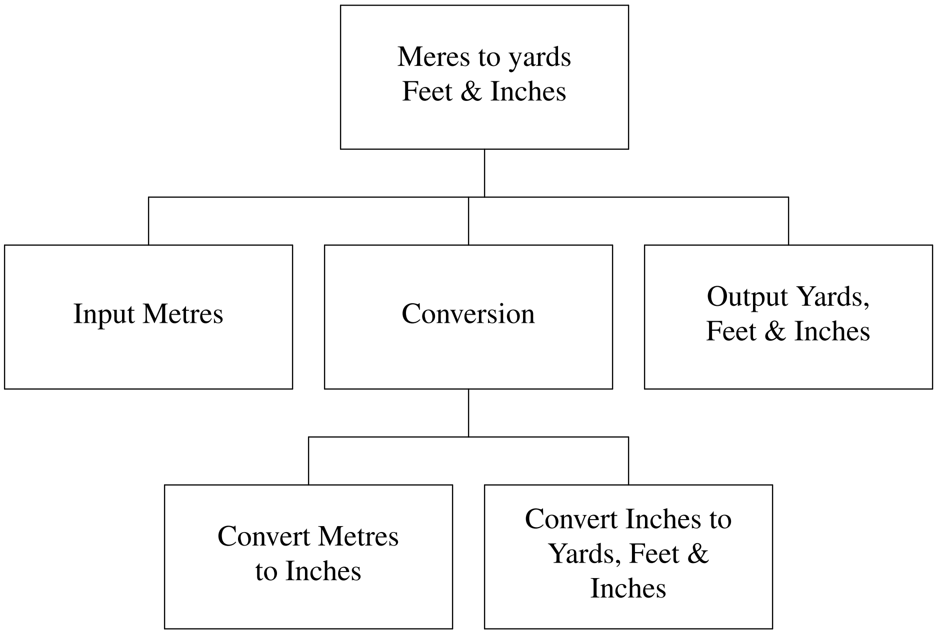

Table 1: *Top down analysis*

We will represent the imperial distance using a user defined type `IMPERIAL_DIST` defined as a three element array of type `NATURAL`. We can then identify the following procedures:

1) `DISTANCE_CONVERSION` (top level procedure): Input, calls to `CONVERT` and `OUTPUT`. Include type definition for `IMPERIAL_DIST`.

NAME	USAGE	TYPE	RANGE
METRES	Input variable	POSITIVE	1..1000000
YFI_DIST	Local variable (array)	IMPERIAL_DIST	n.a.

2) `CONVERT` (level 2 function): Convert to inches and then to imperial units. Return value of type `IMPERIAL_DIST`.

NAME	USAGE	TYPE	RANGE
METRES	Formal parameter	POSITIVE	Default
INCHES	Local variable	FLOAT	Default
YFI	Return variable (array)	IMPERIAL_DIST	n.a.

3) `OUTPUT` (level 2 procedure): Output array result.

NAME	USAGE	TYPE	RANGE
YFI	Formal parameter	IMPERIAL_DIST	n.a.

A complete Nassi-Shneiderman design for the above is given below:

INCHES := METRES*39.37
Determine Yards (YFI(1))
Determine Feet (YFI(2))
Determine Inches (YFI(3))
Return YFI array

CONVERT
(Level 2 function)

Output Yards (YFI_OUTPUT(1))
Output Feet (YFI_OUTPUT(2))
Output Inches (YFI_OUTPUT(3))

OUTPUT
(Level 2 procedure)

Input METRES
YFI_DIST := CONVERT(METRES)
OUTPUT(YFI_DIST)

DISTANCE_CONVERT
(Top level procedure)

Table 2: *Nassi-Shneiderman chart*

3. Implementation

The encoding of the above design is given in Table 3.

```
-- DISTANCE CONVERSION
-- 15 August 1997
-- Frans Coenen
-- Dept Computer Science, University of Liverpool

with TEXT_IO;
use TEXT_IO;

procedure DISTANCE_CONVERT is
        LOWER_BOUND       : constant := 1;
        UPPER_BOUND       : constant := 3;
        CONVERSION_FACTOR : constant := 39.37;
        package FLOAT_INOUT is new FLOAT_IO(FLOAT);
        use FLOAT_INOUT;
        package INTEGER_INOUT is new INTEGER_IO(INTEGER);
        use INTEGER_INOUT;
        type IMPERIAL_DIST is array (LOWER_BOUND..UPPER_BOUND) of NATURAL;
        METRES   : POSITIVE range 1..1000000;
        YFI_DIST : IMPERIAL_DIST;
------------------------------------------------------------
-- CONVERT
        function CONVERT(METRES: POSITIVE) return IMPERIAL_DIST is
                INCHES : FLOAT;
                YFI    : IMPERIAL_DIST;
        begin
-- Metes to inches
                INCHES := FLOAT(METRES)*CONVERSION_FACTOR;
-- Yards
                YFI(1) := INTEGER(INCHES/36.0-0.5);
                INCHES := INCHES - (YFI(1)*36);
-- Feet
                YFI(2) := INTEGER(INCHES/12.0-0.5);
-- Inches
                YFI(3) := INTEGER(INCHES - FLOAT(YFI(2)*12));
                RETURN(YFI);
        end CONVERT;

------------------------------------------------------------
-- OUTPUT
        procedure OUTPUT(YFI_OUTPUT: IMPERIAL_DIST) is
        begin
                PUT(YFI_OUTPUT(LOWER_BOUND));
                PUT(" Yards, ");
                PUT(YFI_OUTPUT(LOWER_BOUND+1));
                PUT(" Feet, ");
                PUT(YFI_OUTPUT(LOWER_BOUND+2));
                PUT_LINE(" Inches");
        end OUTPUT;

------------------------------------------------------------
begin
        PUT_LINE("Input distance in Metres and decimals of Metres:");
        GET(METRES);
        NEW_LINE;
        YFI_DIST := CONVERT(METRES);
        OUTPUT(YFI_DIST);
end DISTANCE_CONVERT;
```

Table 3: *Implementation*

4. Testing

4.1. Black box testing

BVA testing and limit testing: A set of appropriate cases is given in Table 4 and the result in Table 5.

TEST CASE DECIMAL_NUMBER	EXPECTED RESULT OUTPUT
0	CONSTRAINT_ERROR
1	{1, 0, 3}
2	{2, 0, 7}
999999	{1093610, 0, 1}
1000000	{1093611, 0, 4}
1000001	CONSTRAINT_ERROR

Table 4: BVA and Limit Test Cases

```
kuban-352 $ distance_convert
Input distance in Metres and decimals of Metres:
0
Ada-runtime: Exception CONSTRAINT_ERROR raised in
/tmp_mnt/home/staff2/ra/frans/public_html/2CS21/week10/distance_convert.ada
on line 50. Rangecheck error: 0 not in range 1..1000000.

kuban-353 $ distance_convert
Input distance in Metres and decimals of Metres:
1
1 Yards, 0 Feet, 3 Inches

kuban-354 $ distance_convert
Input distance in Metres and decimals of Metres:
2
2 Yards, 0 Feet, 7 Inches

kuban-355 $ distance_convert
Input distance in Metres and decimals of Metres:
999999
1093610 Yards, 0 Feet, 1 Inches

kuban-356 $ distance_convert
Input distance in Metres and decimals of Metres:
1000000
1093611 Yards, 0 Feet, 4 Inches

kuban-357 $ distance_convert
Input distance in Metres and decimals of Metres:
1000001
Ada-runtime: Exception CONSTRAINT_ERROR raised in
/tmp_mnt/home/staff2/ra/frans/public_html/2CS21/week10/distance_convert.ada
on line 50. Rangecheck error: 1000001 not in range 1..1000000.
```

Table 5: Results of BVA and Limit Testing

Arithmetic testing: We should include test cases to ensure correct operation of arithmetic expressions. In this case it is appropriate to test each such expression with positive and zero sample values where applicable. Some of this is taken in to consideration by the above BVA and Limit tests. However, we should include a test where a positive number of feet result (rather than zero). An appropriate test case is given in Table 6 below and the result in Table 7.

TEST CASE DECIMAL_NUMBER	EXPECTED RESULT OUTPUT
4	{4, 1, 1}

Table 6: BVA and Limit Test Cases

```
kuban-360 $ distance_convert
Input distance in Metres and decimals of Metres:
4
4 Yards, 1 Feet, 1 Inches
```

Table 7: Results of Arithmetic Testing