EXAMPLE PROGRAM FACTORIAL

1. Requirements

Design and implement an Ada program that returns the value of N! (N *Factorial/fR) given a specific value for N. Note:*

$$n! = n \times (n-1) \times (n-2) \ldots 3 \times 2 \times 1$$

For example:

$$10! = 10 \times 9 \times 8 \times 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1 = 3628800$$

Note also that N must be a positive integer and assume that the maximum permissible value for N is 12.

2. Design

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

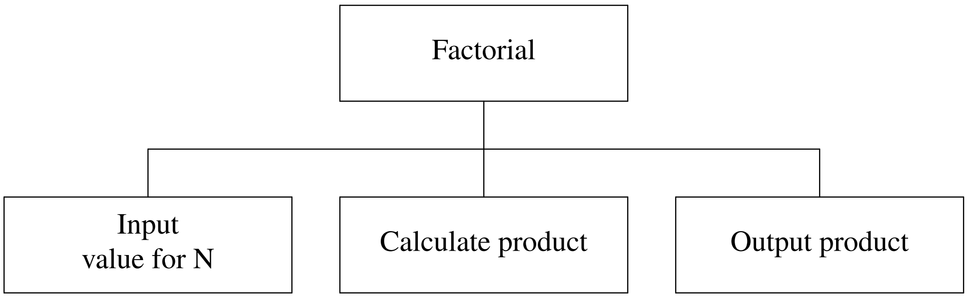

Table 1: *Top down analysis*

The operations given in Table 1 are trivial enough to be implemented as a single procedure incorporating a single fixed count loop.

`FACTORIAL` (top level procedure). Input number of terms and calculate product.

Name	Description	Type	Range
NUMBER_OF_TERMS	Global input variable	POSITIVE	1..12
COUNTER	Global variable	POSITIVE	Default

`FACTORIAL` (top level procedure).

So as to adhere to the generally accepted mathematical definition of *factorial(N)* we will process the loop in reverse. An appropriate Nassi-Shneiderman chart is given in Table 2.

Input NUMBER_OF_TERMS
PRODUCT := NUMBER_OF_TERMS
Fixed count loop COUNTER 1 to NUMBER_OF_TERMS-1 (reverse)
PRODUCT := PRODUCT*COUNTER
output PRODUCT

Table 2: Nassi-Shneiderman chart

3. Implementation

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

```
-- FACTORIAL
-- 15 September 1997
-- Frans Coenen
-- Dept Computer Science, University of Liverpool

with TEXT_IO;
use TEXT_IO;

procedure FACTORIAL is
        package INTEGER_INOUT is new INTEGER_IO(INTEGER);
        use INTEGER_INOUT;
        NUMBER_OF_TERMS : POSITIVE range 1..12;
        PRODUCT         : POSITIVE;
begin
-- Input N_VALUE
        PUT_LINE("Input number of terms (integer 1..12): ");
        GET(NUMBER_OF_TERMS);
        PRODUCT := NUMBER_OF_TERMS;
-- For loop
        for COUNTER in reverse 1..NUMBER_OF_TERMS-1  loop
                PRODUCT := PRODUCT*COUNTER;
        end loop;
-- Output result
        PUT(NUMBER_OF_TERMS);
        PUT("! = ");
        PUT(PRODUCT);
        NEW_LINE;
end FACTORIAL;
```

Table 3: *Implementation*

4. Testing

4.1. Black box testing

BVA, Limit and Arithmetic testing. A suitable set of test cases is given in the Table 4 and the results in table 5.

TEST CASE	EXPECTED RESULT
NUMBER_OF_TERMS	OUTPUT
0	CONSTRAINT_ERROR
1	1
2	2
6	720
11	39916800
12	479001600
13	CONSTRAINT_ERROR

Table 4: *BVA, Limit and Arithmetic test cases.*

```
kuban-421 $ factorial
Input number of terms (integer 1..12):
0
Ada-runtime: Exception CONSTRAINT_ERROR raised in
/tmp_mnt/home/staff2/ra/frans/public_html/2CS21/week7/factorial.ada
on line 17. Rangecheck error: 0 not in range 1..12.

kuban-422 $ factorial
Input number of terms (integer 1..12):
1
          1! =           1

kuban-423 $ factorial
Input number of terms (integer 1..12):
2
          2! =           2

kuban-424 $ factorial
Input number of terms (integer 1..12):
6
          6! =         720

kuban-425 $ factorial
Input number of terms (integer 1..12):
11
         11! =    39916800

kuban-426 $ factorial
Input number of terms (integer 1..12):
12
         12! =   479001600
```

Table 5: *Result of BVA, Limit and Arithmetic testing*

```
kuban-427 $ factorial
Input number of terms (integer 1..12):
13
Ada-runtime: Exception CONSTRAINT_ERROR raised in
/tmp_mnt/home/staff2/ra/frans/public_html/2CS21/week7/factorial.ada
on line 17. Rangecheck error: 13 not in range 1..12.
```

Table 5 Continued

4.2. White box testing

Loop testing. A suitable set of test cases is given in the Table 6. Except for the third test case ("two passes through loop") all these have been run as either BVA, limit or arithmetic test cases (see Table 4). The result of running the third test case is given in Table 7.

TEST CASE	EXPECTED RESULT
NUMBER_OF_TERMS	OUTPUT
1 (no passes through loop)	1
2 (one pass through loop)	2
3 (two passes through loop)	6
6	720
11 (end value -1 passes)	39916800
12 (end value passes)	479001600

Table 6: *Loop test cases.*

```
kuban-431 $ factorial
Input number of terms (integer 1..12):
3
          3! =           6
```

Table 7: *Result of loop tests*

4.3. Data validtaion tersting

Resultys of a number of data validation test are given in Table 8.

```
kuban-312 $ factorial
Input number of terms (integer 1..12):
frans
Ada-runtime: Exception DATA_ERROR propagated out of main.

kuban-313 $ rans
ksh: rans:  not found

kuban-314 $ factorial
Input number of terms (integer 1..12):
4.0
          4! =          24

kuban-315 $ 0
ksh: 0:  not found

kuban-316 $ factorial
Input number of terms (integer 1..12):
1s0
Ada-runtime: Exception DATA_ERROR propagated out of main.

kuban-317 $ 0
ksh: 0:  not found
```

Table 8: *Result of data validation tests*