EXAMPLE PROGRAM ASCII CODE OUTPUT

1. Requirements

Produce a program that outputs the ASCII character set from 4 to 127 (the first few ASCII codes are control characters which can not be handled by many editors/word processors and thus we have excluded them from this exercise).

2. Design

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

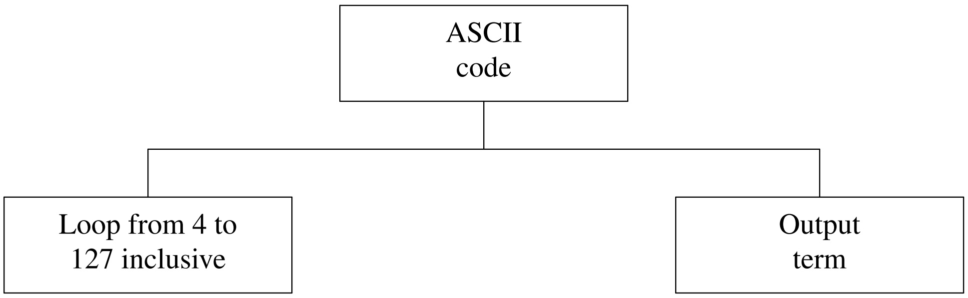

Table 1: *Top down analysis.*

This can be implemented using a single fixed count loop embedded in a single (top level) procedure. We will include a counter that is incremented on each loop and causes a `NEW_LINE` to be output when it reaches 4 (after which it will be reset to 1).

1) `ASCII_CODE` (top level procedure): Loop through ascii codes from 4 to 127 and output characters.

Name	Description	Type	Range
START_COND	Global constant	INTEGER	4
END_COND	Global constant	INTEGER	127
COUNTER	Global variable	INTEGER	1..6

An appropriate Nassi-Shneiderman chart is given in Table 2 and a flow chart in Table 3.

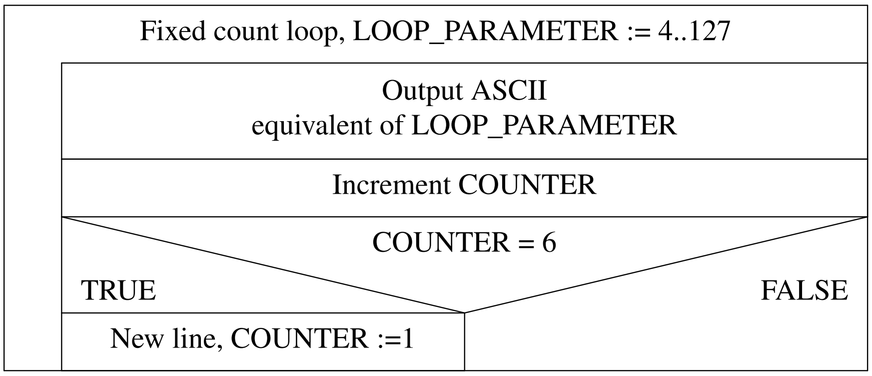

Table 2: *Nassi-Shneiderman chart*

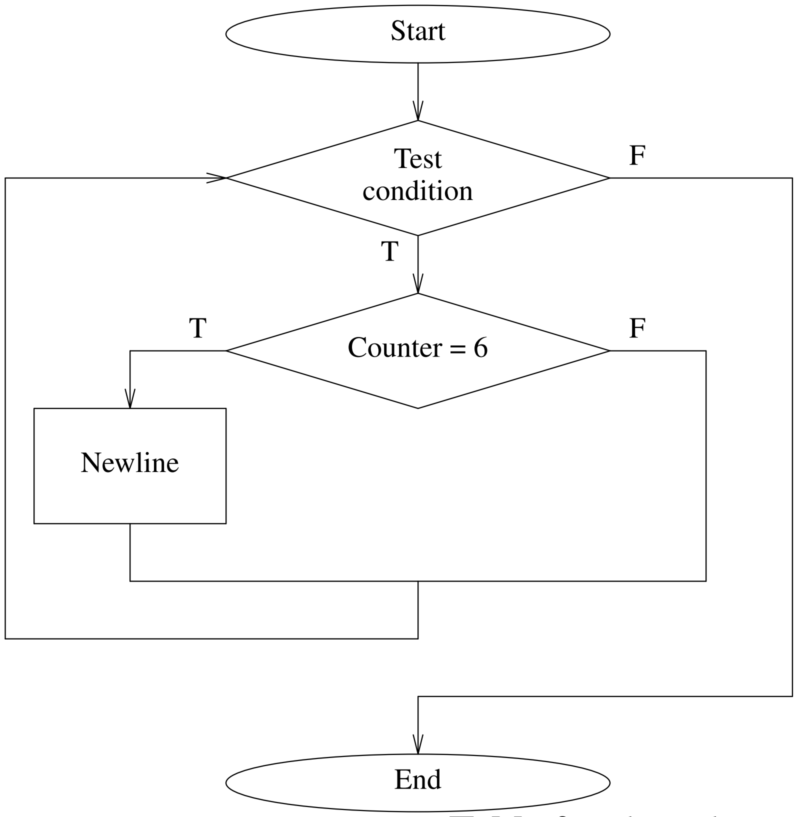

Table 2: *Flow chart*

3. Implementation

The encoding of the above design is given in Table 4.

```
-- ASCII CODE
-- 16 August 1997
-- Frans Coenen
-- Dept Computer Science, University of Liverpool

with TEXT_IO;
use TEXT_IO;

procedure ASCII_CODE is
    package INTEGER_INOUT is new INTEGER_IO(INTEGER);
    use INTEGER_INOUT;
    START_COND : <B>constant</B>       := 4;
    END_COND   : <B>constant</B>       := 127;
    COUNTER    : INTEGER is range 1..6 := 1;

begin
    for LOOP_PARAMETER in START_COND..END_COND loop
        PUT(LOOP_PARAMETER, WIDTH=>3);
        PUT(" = ");
        PUT(CHARACTER'VAL(LOOP_PARAMETER));
        COUNTER := COUNTER+1;
        if COUNTER = 6 then
            NEW_LINE;
            COUNTER := 1
        end if;
    end loop;
end ASCII_CODE;
```

Table 4: *Implementation*

4. Testing

The above code contains no input, no ranged data items and no arithmetic expressions. The loop is a fixed count loop and should be exercised with a complete run through. This will be the only testing that is required. The result is given in Table 5.

```
kuban-248 $ ascii_code
      4 =        5 =        6 =        7 =        8 =
      9 =            10 =
     11 =       12 =       13 =
     14 =       15 =       16 =       17 =       18 =
     19 =       20 =       21 =       22 =       23 =
     24 =       25 =       26 =       27 =      28 =
     29 =       30 =       31 =       32 =        33 = !
     34 = "      35 = #      36 = $      37 = %      38 = &
     39 = '      40 = (      41 = )      42 = *      43 = +
     44 = ,      45 = -      46 = .      47 = /      48 = 0
     49 = 1      50 = 2      51 = 3      52 = 4      53 = 5
     54 = 6      55 = 7      56 = 8      57 = 9      58 = :
     59 = ;      60 = <      61 = =      62 = >      63 = ?
     64 = @      65 = A      66 = B      67 = C      68 = D
     69 = E      70 = F      71 = G      72 = H      73 = I
     74 = J      75 = K      76 = L      77 = M      78 = N
     79 = O      80 = P      81 = Q      82 = R      83 = S
     84 = T      85 = U      86 = V      87 = W      88 = X
     89 = Y      90 = Z      91 = [      92 = \      93 = ]
     94 = ^      95 = _      96 = `      97 = a      98 = b
     99 = c     100 = d     101 = e     102 = f     103 = g
    104 = h     105 = i     106 = j     107 = k     108 = l
    109 = m     110 = n     111 = o     112 = p     113 = q
    114 = r     115 = s     116 = t     117 = u     118 = v
    119 = w     120 = x     121 = y     122 = z     123 = {
    124 = |     125 = }     126 = ~     127 =
```

Table 4: *Test output*

Note that some of the character codes are unprintable. Note also that:

9	=	Horizintal tab
10	=	Line feed/carriage return
27	=	Escape
32	=	Space
127	=	Delete