

```

8:     return  $N_n$ 
9: end if

10:   $t \leftarrow 1$  ▷ Token number  $t$ 
11:  repeat
12:     $type \leftarrow \text{READUINT8}(B_{t, \text{TYPE}})$ 
13:    if  $type = \text{CHAR}$  then
14:       $T_{n,t} \leftarrow \text{READCHAR}(B_{t, \text{CHAR}})$ 
15:    else if  $type = \text{STRING}$  then
16:       $T_{n,t} \leftarrow \text{READSTRING}(B_{t, \text{STRING}})$ 
17:    else if  $type = \text{DIGITS}$  then
18:       $T_{n,t} \leftarrow \text{READUINT32}(B_{t, \text{DIGITS}})$ 
19:    else if  $type = \text{DIGITS0}$  then
20:       $d \leftarrow \text{READUINT32}(B_{t, \text{DIGITS0}})$ 
21:       $l \leftarrow \text{READUINT8}(B_{t, \text{DZLEN}})$ 
22:       $T_{n,t} \leftarrow \text{LEFTPADNUMBER}(d, l)$ 
23:    else if  $type = \text{DELTA}$  then
24:       $T_{n,t} \leftarrow T_{m,t} + \text{READUINT8}(B_{t, \text{DELTA}})$ 
25:    else if  $type = \text{DELTA0}$  then
26:       $d \leftarrow T_{m,t} + \text{READUINT8}(B_{t, \text{DELTA0}})$ 
27:       $l \leftarrow \text{LENGTH}(T_{m,t})$  ▷ String length including leading zeros
28:       $T_{n,t} \leftarrow \text{LEFTPADNUMBER}(d, l)$ 
29:    else if  $type = \text{MATCH}$  then
30:       $T_{n,t} \leftarrow T_{m,t}$ 
31:    else
32:       $T_{n,t} \leftarrow ''$ 
33:    end if
34:     $N_n \leftarrow N_n \# T_{n,t}$ 
35:     $t \leftarrow t + 1$ 
36:  until  $type = \text{END}$ 
37:  return  $N_n$ 
38: end function

```

Given a complex name with both position and type specific values, this can lead to many separate data streams. The name tokeniser codec is a format within a format, as the multiple byte streams $B_{pos,type}$ are serialised into a single byte stream.

The serialised data stream starts with two unsigned little ~~endian~~-endian 32-bit integers holding the total size of uncompressed name buffer and the number of read names. This is followed the array elements themselves. Note the uncompressed size is calculated as the sum of all name lengths including a termination byte per name (e.g. the nul char). This is irrespective of whether the implementation produces data in this form or whether it returns separate name and name-length arrays.

Token types, $ttype$ holds one of the token ID values listed above in the list above, plus special values to indicate certain additional flags. Bit 6 (64) set indicates that this entire token data stream is a duplicate of one earlier. Bit 7 (128) set indicates the token is the first token at a new position. This way we only need to store token types and not token positions.

The total size of the serialised stream needs to be already known, in order to determine when the token types finish.