```
        return $N_n$
    end if

    $t \leftarrow 1$                                              ▷ Token number $t$
    repeat
        $type \leftarrow$ ReadUint8($B_{t,\texttt{TYPE}}$)
        if $type =$ CHAR then
            $T_{n,t} \leftarrow$ ReadChar($B_{t,\texttt{CHAR}}$)
        else if $type =$ STRING then
            $T_{n,t} \leftarrow$ ReadString($B_{t,\texttt{STRING}}$)
        else if $type =$ DIGITS then
            $T_{n,t} \leftarrow$ ReadUint32($B_{t,\texttt{DIGITS}}$)
        else if $type =$ DIGITS0 then
            $d \leftarrow$ ReadUnt32($B_{t,\texttt{DIGITS0}}$)
            $l \leftarrow$ ReadUint8($B_{t,\texttt{DZLEN}}$)
            $T_{n,t} \leftarrow$ LeftPadNumber($d,\ l$)
        else if $type =$ DELTA then
            $T_{n,t} \leftarrow T_{m,t} +$ ReadUint8($B_{t,\texttt{DELTA}}$)
        else if $type =$ DELTA0 then
            $d \leftarrow T_{m,t} +$ ReadUint8($B_{t,\texttt{DELTA0}}$)
            $l \leftarrow$ Length($T_{m,t}$)              ▷ String length including leading zeros
            $T_{n,t} \leftarrow$ LeftPadNumber($d,\ l$)
        else if $type =$ MATCH then
            $T_{n,t} \leftarrow T_{m,t}$
        else
            $T_{n,t} \leftarrow$ ‘’
        end if
        $N_n \leftarrow N_n \mathbin{+\!\!+} T_{n,t}$
        $t \leftarrow t + 1$
    until $type =$ END
    return $N_n$
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 ~~endiand~~ endian 32-bit integers holding the total size of uncompressed name buffer and the number of read names, and a flag byte indicating whether data is compressed with arithmetic coding or rANS Nx16. 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. ~~This is followed the array elements themselves.~~

This is then followed by serialised data and meta-data for each token stream. 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.