Huffman Code (C) 2023 by Karl Kechele

A **Huffman Code** is:
- a prefix code: no code is a prefix of another code
- optimal: shortest possible code, best compression of data
- depends on the probability distribution of symbols
- used for lossless data compression

In the app you can see 3 tabs and 4 buttons:
- a tab ‚Text‘ with the original text
- a tab ‚Huffman Code‘ with the generated Huffman code and the code tree
- a tab ‚Encoded Text‘ with the encoded text
- a button ‚Encode‘ to encode the text in tab ‚Text‘
- a button ‚Decode‘ to decode the text in tab ‚Encoded Text‘
- a button ‚Copy‘ to copy the active tab to clipboard
- a button ‚New‘ to clear all tabs

In the app you can do:
- edit a text in ‚Text‘ and click ‚Encode‘ to encode the text
- edit the encoded text in ‚Encoded Text‘ and click ‚Decode‘ to decode the text
- change the text in ‚Text‘ and encode it again with the last Huffman coding by clicking ‚Encode‘
- delete the Huffman coding by clicking ‚Clear‘ in the ‚Huffman Code‘-tab
  -> the Huffman code will be generated again after you click ‚Encode‘ next time
- have a look for the Huffman code in tab ‚Huffman Code‘:
  first you can see the codes of the characters, then the distribution and finally the coding tree
  Codes: ‚char‘=<bitcode>
  Distribution: (can be used in ‚Text‘ for only get the Huffman code)
  ;=ch1,ch2,ch2,… -> list of characters
  n1,n2,n3,… -> list of counts of the characters above
  Example: line 1: ;=a,b,c,d line 2: 12,4,25,9 means: 12x a, 4x b, 25x c, 9x d
  Code Tree: #node •(weight) : 0.#next 1.#next -> a node with following nodes #next
  or #node char(weight) : -> an end node for a character ‚char‘
  weight : count of occurrence

See also the examples on the following site.

The rate will be calculated:

rate : count of necessary bits at Huffman coding in relation to a 7bit-ASCII coding
rate = <count of Huffman coding bits> / <count of 7bit-ASCII bits> (in percent)
<count of Huffman coding bits> : count of bits in tab ‚Encoded Text‘
<count of 7bit-ASCII bits> = <count of characters in tab ‚Text‘> * 7 (bits per character)

Example 1: normal usage

Edit in ‚Text‘:
this is an example text

Press ‚Encode‘ and have a look for the ‚Huffman Code‘:

```
;Huffman Code (1.5) (C) 2023 by Karl Kechele
;'char'=Bitcode
'x'=000
'h'=0010
'n'=0011
'm'=0100
'p'=0101
't'=011
'e'=100
'l'=1010
'i'=1011
' '=110
's'=1110
'a'=1111
----------
;Distribution
;=x,h,n,m,p,t,e,l,i, ,s,a
2,1,1,1,1,3,3,1,2,4,2,2
----------
;Code Tree: #node •(weight) : 0.#next 1.#next
;or           #node char(weight) :
#22 •(23) : 0.#20 1.#21
#20 •(9) : 0.#16 1.#17
#16 •(4) : 0.#8 1.#12
#8 x(2) :
#12 •(2) : 0.#0 1.#1
#0 h(1) :
#1 n(1) :
…
```

see in ‚Encoded Text‘:

```
01100101011111011010111110110111100111101000001111010001011010100110011100000011
```

Example 2: define a distrubution (only for generating a Huffman Code)

Edit in ‚Text‘: (the definition means: a text with 12x a, 4x b, 25x c, 9x d)

```
;=a,b,c,d
 12,4,25,9
```

Press ‚Encode‘ and have a look for the ‚Huffman Code‘