⊞— CreateCallout Function

⊞— Word Wrapping Function

⊞— BMP Matrix without read the file

Arial

in the Arial array shown in the above Area there are:
- 6 rows of 20 pixel each + one row of 1 pixel I guess at the bottom of the area/picture
- 16 columns (Character) of 14 pixel each + one row of 1 pixel I guess at the extreme right of the area/picture

Character size - 14 width x 20 height (in pixel)

16 Columns x 6 Rows

$rows(Arial) = 121$ $cols(Arial) = 225$

Attention some Character are not properly positioned in the Array so it is required a refinement of this aspect

$C\$:= \text{" !"\#\$\%\&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz\{}$

Generating your own font

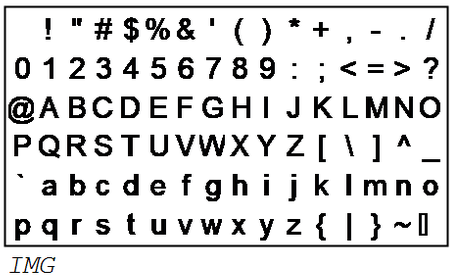
$$d2h_2(x) := \left| \begin{array}{l} h(n) := \text{substr}("0123456789ABCDEF", \text{trunc}(n)+1, 1) \\ \text{concat}\left(h\left(\frac{x-\text{mod}(x,16)}{16}\right), h(\text{mod}(x,16))\right) \end{array} \right.$$

```
C$ := ""      for k ∈ [0..94] + 32
                C$ := concat(C$, "\00", d2h2(k), "\")
```

You can add or quit characters here.

$C\$ = \text{" !"\#\$\%\&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxy}$

$IMG := \text{excel\_EMF}("SubmtrxChrRef.xlsx\ ", "Font", "A1:P6", 0) = "clipboard\_EMF|Option\#0"$



IMG

I don't know how to convert this into a matrix, without saving it from this component and reloading, pasting the numbers, etc. We go to use the Oscar's Arial bitmap.

Conversion of a Text String in a bitmap string built with a monospaced (bitmap) font

Scalable monospaced bitmapped fonts in XYPlot

$CWD := \text{CurrentDirectory}("")$

$\text{CurrentDirectory}(CWD) = "F:\USERS\Desktop\"$

$SubmtrxRef := \text{importData}_{XLSX}("SubmtrxChrRef.xlsx\ ", "sheet1", "A2", "E97")$

$$SubmtrxRef = \begin{bmatrix} 1 & 1 & 20 & 1 & 14 \\ 2 & 1 & 20 & 15 & 28 \\ & & \vdots & & \end{bmatrix}$$

Load from an Excel file a matrix of data, each row is related to a Character in the string C$, indeed the 5 columns have the following meaning:

Column1: is the position of a character in C$ (for instance the number 4 corresponds to the character "#", space " " is the 1st character of the string)
Columns from 2 to 5 are the rows and columns index for the extraction of a submatrix from the Arial matrix corresponding to the character whose position is specified to the first column of this matrix , in details they are:

Column2: is the index of the row start range
Column3: is the index of the row end range

Column4: is the index of the column start range
Column5: is the index of the column end range

```
Path := concat(DocumentDirectory(""), "Arial.bmp") = "C:\Users\DELL\Downloads\XY Plot Examples\___Ese
Arial := READBMP(Path)
```

Text to BMP conversion function

Purpose of the user defined function : converts each character of a text string in a BMP taken from a specified font family stored in a matrix (Monospaced)

Input : TextMatrix = matrix with in each rows a text string to convert in BMP,
FontFamily = matrix that stores BMP repesentation of each char in C$

Output:Text_BMP = matrix that strores in each row the result of the conversion of the corresponding Input row of the matrix TextMatrix in BMP

```
text2BMP(TextMatrix, FontFamily) :=
    CharH := 20
    N_Rows := rows(TextMatrix)
    for r ∈ [1..N_Rows]
        TextString := TextMatrix_r
        τ := matrix(CharH, 0)
        for c ∈ [1..strlen(TextString)]
            Char := substr(TextString, c, 1)
            Char_Index := max(stack(findstr(C$, Char), 0))
            if Char_Index = 0
                τ := augment(τ, submatrix(Arial, 1, 20, 1, 14))
            else
                τ := augment(τ, submatrix(Arial, SubmtrxRef_Char_Inde
        Text_BMP_r := τ
    Text_BMP
```

I have changed this routine despite the original Alvaro's one,this to use the submatrix function that looked to me useful for the purposes of the function itself . The output is assembled in a different way and is an Array in which in each row there is a BMP array that is the result of the conversion of the corresponding row of text of the Input Array. So it is not produced a single array with all the text rows merged together as a block of text.

$$pos(X, PX, k) := stack\left(X_1, X_2, \frac{cols(PX)}{k \cdot 20}, \frac{rows(PX)}{k \cdot 14}\right)$$

Varous Examples

An attempt to define the size of a character of the selected BItmap Monospaced font (Arial)

in the pos function there is a scale factor that changes the size of the character drawn , have drawn a character with different scale factor to understand the relationship of the character size with the scale factor
From the tests done we see that :

Character height = 1.43 (20 Pixel) for a scale factor k of 1
Character width = 0.70 (14 Pixel) **both are rough evaluations**

reducing the scale factor of $\frac{1}{2}$ or $\frac{1}{3}$ reduces the dimensions of the character matrix of the same quantity in proportion

So to position vertically the rows of bitmap converted text we need to define an offset that is related to the size (height) of the character (based on the k factor) .

Assumptions:

We define for now that this **line spacing** is $\frac{1}{10}$ of the character height

and that the **maximum number of text rows** to display in a Callout will be **4 (the hypotesis is of Callout not too verbose)**

If the number of rows is 1, this...

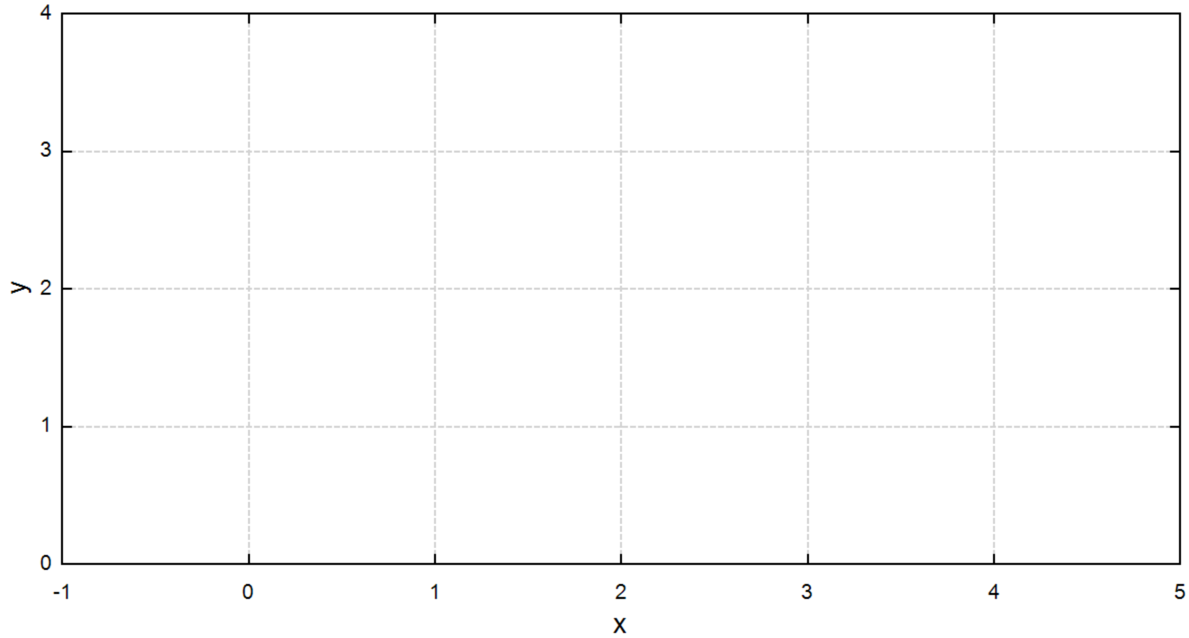
If the number of rows is 1 this row is positioned in 2nd position
2 these rows are positioned in 2nd and 3rd position
3 these rows are positioned in 1st,2nd and 3rd position   see picture on the right for row positions
4 these rows are positioned in 1st,2nd,3rd and 4th position

Size in pixel of drawn Charactesr = 66 x 99 k = 1 - 33 x 49 k = 2 - 22 x 34 k = 3 evaluated using a pixel ruler

$Segment1 := \begin{bmatrix} 0 & 1 \\ 3 & 1 \end{bmatrix}$ $Char\_test := text2BMP\left(\left[\text{"W"} \right], Arial\right)_1$

$Segment2 := \begin{bmatrix} 0 & 2.43 \\ 3 & 2.43 \end{bmatrix}$ $Segment3 := \begin{bmatrix} 0.70 & 1 \\ 0.70 & 3 \end{bmatrix}$ $Segment2s := \begin{bmatrix} 0 & 1.72 \\ 3 & 1.72 \end{bmatrix}$ $Segment3s := \begin{bmatrix} 1.35 & 1 \\ 1.35 & 3 \end{bmatrix}$ Segmer

the coordinates of the segments have been found by **trial and error**



we see that the "W" is n
how he produced the Ari
similar errors in the futur

```
[ [ stack("image:{Char_test}", [pos([0 1], Char_test, 1)], "black", "line", 1) ]
  [ stack("image:{Char_test}", [pos([1 1], Char_test, 2)], "black", "line", 1) ]
  [ stack("image:{Char_test}", [pos([2 1], Char_test, 3)], "black", "line", 1) ] ]
{ Segment1
  Segment2
  Segment3
  Segment2s
  Segment3s
  Segment2ss
  Segment3ss
```

Tests for positioning the bitmaped text on the Callout

```
Text_ := "c'era una volta un testo parecchio lungo e ci si chiedeva se riusciva a stare in una varia
Text_ := "c'era una volta un testo parecchio lungo e ci si chiedeva se riusciva a stare in una varia
Text_ := "c'era una volta un testo parecchio lungo e ci si chiedeva se riusciva a stare in "

Text_ := "c'era una volta un testo parecchio lungo"
```

`TextWidth := 40`   Wrapping length in characters

```
TextRows := WordWrap(Text_, TextWidth) = [ "c'era una volta un testo parecchio lungo"
                                           "e ci si chiedeva se riusciva a stare in " ]
```

```
255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 2
255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 2
255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 2
```

$TxtxCallout := text2BMP(TextRows, FontFamily) =$

$Char\_h := 1.43$ $Char\_w := 0.7$ for k_ equal 1 $k\_ := 3$ this time for these tests

$Row\_offset := \frac{1}{10} \cdot Char\_h$ $Text\_Length := \frac{Char\_w}{k\_} \cdot TextWidth = 9.33$

$Callout1 := CreateCallout(stack([0 0], [1 2]), "blue", 1, 3, "solid")$

$Callout1_{2_{3\,1}} = 1$ $Callout1_{2_{3\,2}} = 2$ Callout "Center" coordinates from the previous definition

$$VertPos := \begin{bmatrix} Callout1_{2_{3\,1}} & Callout1_{2_{3\,2}} + \frac{Row\_offset}{2} \\ Callout1_{2_{3\,1}} & Callout1_{2_{3\,2}} + \frac{Row\_offset}{2} \\ Callout1_{2_{3\,1}} & Callout1_{2_{3\,2}} + \frac{3}{2} \cdot Row\_offset + \frac{Char\_h}{k\_} \\ Callout1_{2_{3\,1}} & Callout1_{2_{3\,2}} + \frac{3}{2} \cdot Row\_offset + \frac{Char\_h}{k\_} \end{bmatrix}$$

Matrix that stores the coordinates of the starting vertical positions of the row(s) of text of the callout, defined according to the specs stated previously. For now I have focused the attention **only** on the vertical positions. In a second moment will see to develop a rule for the horizontal position (an idea is to use the golden ratio of the text wrapping length)

$r\_TextR := rows(TextRows)$

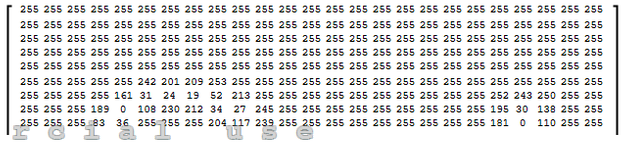
for $i\_rows \in [1..r\_TextR]$

$BMP_{Name} := concat("BMPxCallout", num2str(i\_rows))$

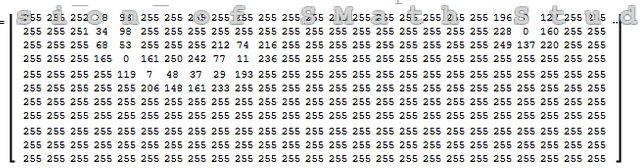
$dummy := str2num(concat(BMP_{Name}, ":", num2str(TxtxCallout_{i\_rows})))$

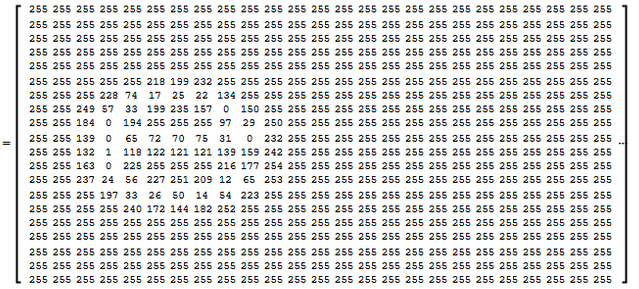
$CalloutStr_{i\_rows} := \left[stack\left(concat("image:{", BMP_{Name}, "}"), \left[pos\left(\left[VertPos_{r\_TextR\,1} \; VertPos_{r\_Te} \right. \right. \right. \right.$

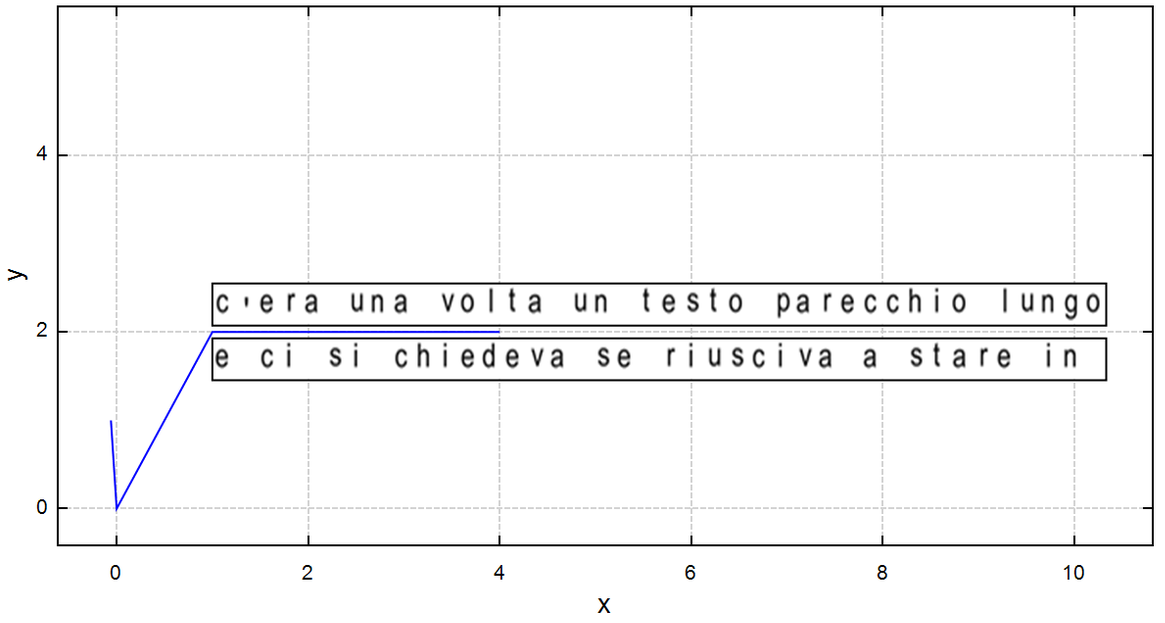
$$\left[\left[\left["image:\{BMPxCallout1\}" \right] \right] \right] \quad \begin{bmatrix} 1 \\ 2.0715 \\ 9.3333 \end{bmatrix}$$



$$CalloutStr = \begin{bmatrix} \begin{bmatrix} \begin{bmatrix} 0.4762 \end{bmatrix} \\ \text{"black"} \\ \text{"line"} \\ 1 \end{bmatrix} \end{bmatrix} \\ \begin{bmatrix} \begin{bmatrix} \text{"image:\{BMPxCallout2\}"} \\ \begin{bmatrix} 1 \\ 1.4518 \\ 9.3333 \\ 0.4762 \end{bmatrix} \\ \text{"black"} \\ \text{"line"} \\ 1 \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

$BMPxCallout1 =$ 

$BMPxCallout2 =$ 



$$\begin{bmatrix} [\text{Callout1}] \\ \text{CalloutStr}_1 \\ \text{CalloutStr}_2 \end{bmatrix}$$