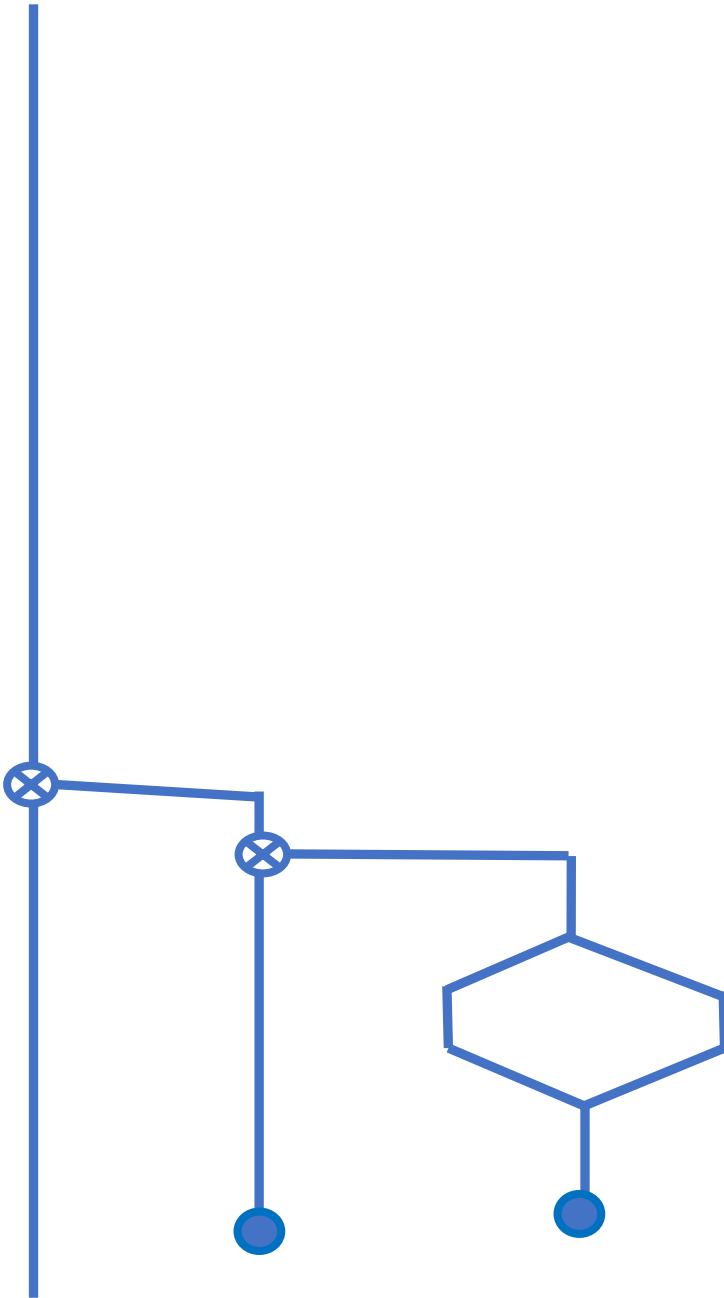


# A Two-Dimension Data Structure

Silicon Valley Forth  
Interest Group  
Sep. 25, 2021  
Bill Ragsdale



# Forth Data Structures

Forth has a variety of data structures. But they exist for its internal operation.

# Forth Data Structures

Forth has a variety of data structures. But they exist for its internal operation.

Forth's inherent data structures:

Byte/character	_	C@ C!
Word	--	W@ W!
Cell	----	@ !
Float	-----	F@ F!
Array	..._	specific words
Files		Include, Require

# The General Need

Problem oriented data structures are the programmer's responsibility.

Data base

Matrices

Vector graphics

Bit graphics

Sound

Video

AI structures

Simulation structures

# My Need

Intended for math and statistical analysis.

Data loaded from .csv files.

Results saved to .csv files.

Data cell size from bytes (1) to floats (8).

Floats and characters for the present.

Notation from *Scientific Forth* by Dr. Julian Noble.

# What Can This Structure Do?

Load with literal values

Load with random numbers

Read and write data files.

Formatted print

Math on matrix, row, column, sub-area, cell.

Copy and move by row, column, sub-area, cell.

Multicolumn sort

Linear algebra operations

Transpose, matrix multiply, equation solution

Inversion, Gaussian elimination, etc.

Limited alphabetic processing

# A Quick Demonstration

```
1 4 create{ x{  
x{ {[ 1 2 3 4 ]}  
x{ }list
```

1.00 2.00 3.00 4.00

```
x{ 4 4 }resize  
x{ }transpose  
x{ }list
```

# A Quick Demonstration

```
1 4 create{ x{  
x{ <[ 1 2 3 4 ]>      1.00 2.00 3.00 4.00  
x{ }list
```

```
x{ 5 3 }resize  
x{ }transpose  
x{ }list
```

1.00	0.00	0.00
2.00	0.00	0.00
3.00	0.00	0.00
4.00	0.00	0.00
0.00	0.00	0.00

ok



# Components

Data area of 'row x column x cell-size' bytes.

# Components

Data area of 'row x column x cell-size' bytes.

A descriptor holding: f (cellsize), R (rows), C (columns)  
and control pointers.

# Components

Data area of 'row x column x cell-size' bytes.

A descriptor holding: f (cellsize), R (rows), C (columns)  
and control pointers.

A Forth name for each matrix.

# Components

Data area of 'row x column x cell-size' bytes.

A descriptor holding: f (cellsize), R (rows), C (columns) and control pointers.

A Forth name for each matrix.

Words to access matrices, single cells and various row & column combinations.



# The Descriptor

Pointer to its name's PFA

Backlink if resized

Pointer to its data area

Cells size: f (8 bytes for floats)

Number of columns: C

Number of rows: R

```
3 5 create{ x{          x{ }.descriptors
```

```
data located at: 44BB94
```

```
    pfa: 44BB78 i.e. x{
```

```
backlink: 0
```

```
    check: 44BB94
```

```
    size: 8
```

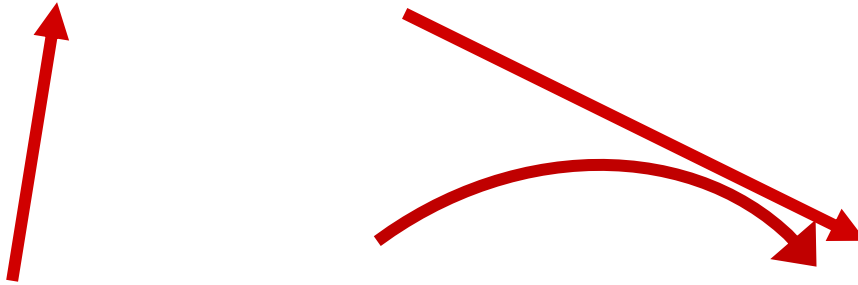
```
params: 5 3 ok
```

# Shown As In Memory

x{ parameter\_field

44BB78 0 4BB94 8 5 3  
pfa link check f C R

8x5x3 bytes  
Data Area



# See The Magic

The word doing the 'heavy lifting' is: `>>`

Used as: `x{ r c }>>`

Where `C` is the matrix number of columns,  
`>>` computes

`address+((row * C) + column) * bytes/cell`

The initial byte address of a float cell.

```
: >> ( x{ r1 c1 --- addr )  
      swap 2pick }cols * +  
      over size: @ * + ; Us
```



# Access A Cell

```
3 4 create{ x{ x{ }integers x{ }list
```




```
0.0 1.0 2.0 3.0  
4.0 5.0 6.0 7.0  
8.0 9.0 10.0 11.0
```

```
x{ 1 1 } F@ F.
```

```
5.0 ok
```

# Row-wise Arithmetic

```
4 3 create{ x{
```

		0.0	0.0	0.0
x{ 1 7e }r#fill		7.0	7.0	7.0
x{ 2 2e }r#fill		2.0	2.0	2.0
x{ 1 x{ 2 x{ 3 }rrr+		9.0	9.0	9.0
x{ }list		ok		

# Matrix Operators

```
create{ create{*  
}list, }sublist  
}copy, }rcopy, }ccopy, }subcopy  
}clear, }integers, }random, }resize  
}dimensions, }rows, }cols, }cells, }bytes  
}det, }transpose, }invert, }bubble
```

# All Math Operators

Matrix	}+	}-	}.*	}/.
Common row	}r+	}r-	}r.*	}r}/.
Independent rows	}rrr+	}rrr-	}rrr.*	}rrr}/.
Common column	}c+	}c-	}c.*	}c}/.
Independent columns	}ccc+	}ccc-	}ccc.*	}ccc}/.
Single cell	}rc+	}rc-	}rc.*	}rc}/.
Common sub-cell	}com+	}com-	}com.*	}com}/.
Independent sub-cells	}sub+	}sub-	}sub.*	}sub}/.

Common root word



(}SubX)

Note: }.\* and }}/. are cell by cell.

# Alphanumerics

```
1 3 15 create{* alpha{  
alpha{ }alphafill  
alpha{ }charlist
```

```
ABCDEFGHIJKLMNO  
PQRSTUVWXYZ[\]^  
_`abcdefghijklm ok
```

# Look Within A Definition

```
0.00  1.00  2.00  3.00  4.00  
5.00  6.00  7.00  8.00  9.00  
10.00 11.00 12.00 13.00 14.00
```

```
x{ }transpose
```

# Look Within A Definition

```
0.00  1.00  2.00  3.00  4.00
5.00  6.00  7.00  8.00  9.00
10.00 11.00 12.00 13.00 14.00
```

```
x{ }transpose
```

```
0.00  5.00  10.0
1.00  6.00  11.0
2.00  7.00  12.0
3.00  8.00  13.0
4.00  9.00  14.0
```

```
ok
```

# Look Within A Word

```
: }Transpose ( x{ --- )
?x{  dup }dimensions opentransient{
dup transient{ }copy
dup params: 2@ swap 2pick params: 2!
dup }dimensions swap
  0 do  dup 0 do
    transient{ i j }}@
      over j i }}!
  loop loop
2drop closetransient{ ;
```

Missing something? Find a model and modify.



# Generate Random Data

```
Matrix 16 x 1      16 1 create{ ref{
Fill random integers  ref{ 10e 0e }random
Matrix 16 x 2      16 2 create{ x{
Fill random floats  x{ 100e 1e }random
Copy integers column 0  ref{ x{ 0 }cCopy
List the composite  x{ }list
```

0.00	4.3
7.00	69.4
5.00	79.9
2.00	9.6
5.00	41.1
0.00	44.5
0.00	96.2
8.00	77.1
4.00	26.8
1.00	43.4
0.00	88.8
2.00	16.7
3.00	16.7
0.00	57.0
0.00	79.5
9.00	51.4

# Sort On Column 1 Then Column 0

0.00	4.3
7.00	69.4
5.00	79.9
2.00	9.6
5.00	41.1
0.00	44.5
0.00	96.2
8.00	77.1
4.00	26.8
1.00	43.4
0.00	88.8
2.00	16.7
3.00	16.7
0.00	57.0
0.00	79.5
9.00	51.4

```
x{ 1 }bubble  
x{ 0 }bubble  
  
x{ }list
```

0.00	4.6
0.00	44.5
0.00	57.0
0.00	79.5
0.00	88.8
0.00	96.2
1.00	43.4
2.00	9.6
2.00	16.7
3.00	16.7
4.00	26.8
5.00	41.1
5.00	79.9
7.00	69.4
8.00	77.1
9.00	51.4

# Sort Performance

A bubble sort with comparisons:  $(n^2 + n)/2$

```
1000 3 create{ x{ x{ 1000e 0e }random  
x{ 0 }bubble
```

```
    100 rows      16 msec.  
   1,000 rows    1.9 seconds.  
   5,000 rows   31.8 seconds.  
  10,000 rows   1 min 57 seconds
```

For daily data that covers 27 years.

# Benefits

A very flexible data structure.

A rich suite of support words.

Current words easily modified for other needs.

So, see my [Win32Forth Guide](#) on Github.

# Future Work

Add file support.

Add statistics support.

Possibly, text entries and added number types.

Expand report generation choices.

# Credits

- Andrew McKewan and Tom Zimmer for Win32Forth.
- The European team who updated it in the early 2000s.
- Dr. Julian V. Noble, *Scientific Forth*

# References

- [https://github.com/BillRagsdale/Forth\\_Projects](https://github.com/BillRagsdale/Forth_Projects)
- <https://github.com/BillRagsdale/WIN32Forth-Guide>