



Better Values

Improving the extended Forth value concept



Better Values

Example:

STRINDEX Indexed strings

State of play at EuroForth 2020

```

: STRINDEXCOMP, ( xt--- ) \ Compiling action of a child of STRINDEX
>BODY \ To the address of the data
CLIT, \ Compile that address
]] DUP @ ROT * + CELL+ [[ \ Calculate address of string
OPERATOR TYPE @OFF CASE
  VMOD@ OF \ Default returns address
  VMOD! OF POSTPONE ZMOVE \ Writes to string
  VMODADDR OF \ ADDR also returns address
  VMODSIZE OF ]] IP>NFA NAME> >BODY @ [[ \ SIZEOF returns max length
  BAD-METHOD
ENDCASE
;

: STRINDEXINTERP ( addr--- ) \ Interpret a child of STRINDEX
\ As for STRINDEXCOMP, but not postponed
...
;

: STRINDEXNEW ( arraysize,maxlen--- ) \ Prepare a new indexed string
1+ DUP \ Set the string size (allow for 0 terminator)
SWAP 1+ * ALLOT&ERASE \ Allocate (allow for 0 or 1 based indexing)
['] STRINDEXCOMP, SET-COMPILER \ When a child is being compiled
INTERP> STRINDEXINTERP \ When a child is being interpreted
;

: STRINDEX ( arraysize,maxlen <name>--- ) \ Create an indexed string from inline
CREATE STRINDEXNEW
;

: ZSTRINDEX ( arraysize,maxlen,z$--- ) \ Create an indexed string from a zstring
ZCOUNT ($CREATE) STRINDEXNEW
;

```



Better Values

Good points...

- Logical separation of operations
- No magic numbers
- All values initialised
- Tidy POSTPONES

```

: STRINDEXCOMP, ( xt--- ) \ Compiling action of a child of STRINDEX
>BODY \ To the address of the data
CLIT, \ Compile that address
...
]] DUP @ ROT * + CELL+ [[ \ Calculate address of string
OPERATOR TYPE @OFF CASE
  VMOD@ OF \ Default returns address
  VMOD! OF POSTPONE ZMOVE \ Writes to string
  VMODADDR OF \ ADDR also returns address
  VMODSIZE OF ]] IP>NFA NAME> >BODY @ [[ \ SIZEOF returns max length
  BAD-METHOD
ENDCASE
;

: STRINDEXINTERP ( addr--- ) \ Interpret a child of STRINDEX
\ As for STRINDEXCOMP, but not postponed
...
;

: STRINDEXNEW ( arraysize,maxlen--- ) \ Prepare a new indexed string
1+ DUP \ Set the string size (allow for 0 terminator)
SWAP 1+ * ALLOT&ERASE \ Allocate (allow for 0 or 1 based indexing)
['] STRINDEXCOMP, SET-COMPILER \ When a child is being compiled
INTERP> STRINDEXINTERP \ When a child is being interpreted
;

: STRINDEX ( arraysize,maxlen <name>--- ) \ Create an indexed string from inline
CREATE STRINDEXNEW
;

: ZSTRINDEX ( arraysize,maxlen,z$--- ) \ Create an indexed string from a zstring
ZCOUNT ($CREATE) STRINDEXNEW
;

```

Euroforth 2022



Better Values

Not so good points...

- No index checks
- No string length checks
- Data close to code
- Unlovely SIZEOF

```
: STRINDEXCOMP, ( xt--- ) \ Compiling action of a child of STRINDEX
>BODY
  >BODY \ To the address of the data
  CLET, \ Compile that address
  ]] DUP @ ROT * + CELL+ [[ \ Calculate address of string
  OPERATOR TYPE @OFF CASE
  VMOD@ OF \ Default returns address
  VMOD! OF POSTPONE ZMOVE \ Writes to string
  VMODADDR OF \ ADDR also returns address
  VMODSIZE OF ]] IP->NFA NAME> >BODY @ [[ \ SIZEOF returns max length
  BAD-METHOD
  ENDCASE
;

: STRINDEXINTERP ( addr--- ) \ Interpret a child of STRINDEX
..
\ As for STRINDEXCOMP, but not postponed
..
;

: STRINDEXNEW ( arraysize,maxlen--- ) \ Prepare a new indexed string
1+ DUP , \ Set the string size (allow for 0 terminator)
SWAP 1+ * ALL0T&ERASE \ Allocate (allow for 0 or 1 based indexing)
[!] STRINDEXCOMP, SET-COMPILER \ when a child is being compiled
INTERP> STRINDEXINTERP \ when a child is being interpreted
;

: STRINDEX ( arraysize,maxlen <name>--- ) \ Create an indexed string from inline
CREATE STRINDEXNEW
;

: ZSTRINDEX ( arraysize,maxlen,z$--- ) \ Create an indexed string from a zstring
ZCOUNT (SCREATE) STRINDEXNEW
;
```

Euroforth 2022



Better Values

Existing word for index checking

```
: VICHECK { pindex paddr -- pindex' paddr } \ Checks for valid index
\ paddr is the address of the data, the first cell of which contains the array size
pindex 0 paddr @ WITHIN IF \ Index is valid
  pindex paddr
ELSE \ Index is invalid
  Z" Invalid index " pindex ZFORMAT Z+
  Z" for " Z+ paddr >NAME 1+ Z+ \ >NAME does not work for separated data
  Z" length " Z+ paddr @ ZFORMAT Z+
  ERROR
  0 paddr \ Use zeroth index
THEN
;
```

Could do better...

Euroforth 2022



Better Values

String length checking

```
: SLCHECK ( pz$,pindex,paddr---pz$,pindex,paddr ) \ Check for string length
2 PICK ZCOUNT NIP \ Get length of string
OVER CELL+ @ \ Get maximum length
U> IF \ Overflow
  Z" String length overflow index "
  2 PICK ZFORMAT Z+
  Z" for " Z+ OVER >NAME 1+ Z+ \ >NAME does not work for separated data
  FATAL \ A buffer overflow is fatal
THEN
;
```

Buffer overflow is fatal!

Euroforth 2022



Better Values

Include the checks in the child compilation

```
: STRIADDR ( index,bodyaddr---elementaddr ) \ Calculate address of string element
DUP CELL+ @ 1+ ROT * + 2 CELLS+ \ Optimises to 27 bytes, 8 instructions
;

: STRINDEXCOMP, ( ?,index,xt--- ) \ Compiling action of a child of STRINDEX
>BODY \ To the address of the data
CLIT, \ Compile that address
POSTPONE VICHECK \ Index check
OPERATOR TYPE @OFF CASE
VMOD@ OF POSTPONE STRIADDR ENDOF \ Default returns string address
VMOD! OF ]] SLCHECK STRIADDR ZMOVE [[ ENDOF \ Writes to string
VMODADDR OF POSTPONE STRIADDR ENDOF \ ADDR also returns address
VMODSIZE OF ]] NIP CELL+ @ [[ ENDOF \ SIZEOF returns max length
BAD-METHOD
ENDCASE
;
```

Euroforth 2022



Better Values

Move to separated data part 1

```
: STRINDEXNEW ( arraysize, maxlen --- pdata ) \ Make a new child of STRINDEX
2DUP 1+ SWAP 1+ * 2+ CELLS IRESERVE&ERASE \ Allow for 0 term and 0/1 index
TUCK CELL+ ! \ Set maximum length
TUCK ! \ Set index size
;

: STRINDEX ( arraysize,maxlen <name>--- ) \ Create an indexed string from inline
STRINDEXNEW \ Make a new child of STRINDEX
CREATE , \ Create header and set address of data
['] STRINDEXCOMP, SET-COMPILER \ When a child is being compiled
INTERP> STRINDEXINTERP \ When a child is being interpreted
;
```

Euroforth 2022



Better Values

Move to separated data part 2

```
: SLCHECK ( pz$,pindex,p*addr---pz$,pindex,p*addr ) \ Check for string length
2 PICK ZCOUNT NIP \ Get length of string
OVER @ CELL+ @ \ Get maximum length
U> IF \ Overflow
Z" String length overflow index "
2 PICK ZFORMAT Z+
Z" for " Z+ OVER >NAME 1+ Z+ \ >NAME does not work for separated data
FATAL \ A buffer overflow is fatal
THEN
;

: STRIADDR ( index,bodyaddr---elementaddr ) \ Calculate address of string element
@ DUP CELL+ @ 1+ ROT * + 2 CELLS+
;
```

The double fetch problem

Euroforth 2022



Better Values

Adding the secret cell

```
: IRESERVE&ERASE ( n---addr ) \ Reserve separated space, and erase
CELL+ DUP IRESERVE TUCK SWAP ERASE \ Reserve, allowing for secret cell
CELL+ \ Return address of next cell
;

: STRINDEXNEWA ( arraysize, maxlen --- pdata ) \ Make a new STRINDEX - stage A
2DUP 1+ SWAP 1+ * 2 CELLS+ IRESERVE&ERASE \ Allow for 0 term and 0/1 index
TUCK CELL+ ! \ Set maximum length
TUCK ! \ Set index size
;

: STRINDEXNEWB ( pdata --- ) \ Make a new STRINDEX - stage B
DUP \ Set address of data
LATEST CTRL>NFA SWAP CELL- ! \ Save NFA in secret cell
;

: STRINDEX ( arraysize,maxlen <name>--- ) \ Create an indexed string from inline
STRINDEXNEWA \ Make a new child of STRINDEX - stage A
CREATE \ Create header
STRINDEXNEWB \ Make a new child of STRINDEX - stage B
...
```

Euroforth 2022



Better Values

Using the secret cell

```
: VICHECK ( pindex,paddr---pindex',paddr ) \ Checks for valid index
\ paddr is the address of the data, the first cell of which contains the array size
OVER 0 2 PICK @ WITHIN 0= IF \ Index is invalid
Z" Invalid index " 2 PICK ZFORMAT Z+
Z" for " Z+ OVER CELL- @ Z+ \ Add NFA from extra cell
Z" length " Z+ OVER @ ZFORMAT Z+
ERROR
NIP 0 SWAP \ Use zeroth index
THEN
;

: SLCHECK ( pz$,pindex,paddr---pz$,pindex,paddr ) \ Check for string length
2 PICK ZCOUNT NIP \ Get length of string
OVER CELL+ @ \ Get maximum length
U> IF \ Overflow
Z" String length overflow index "
2 PICK ZFORMAT Z+
Z" for " Z+ OVER CELL- @ Z+ \ Add NFA from extra cell
FATAL \ A buffer overflow is fatal
THEN
;
```

Euroforth 2022



Better Values

To do.....

Instead of throwing a FATAL on buffer overflow, log an ERROR and truncate the string before saving.