MUMPS

The Language

History

History 1960s

- Developed at Massachusetts General Hospital in 1966

- **M** assachusetts General Hospital
  U tility
  M ulti
  P rogramming
  S ystem

History 1960s

- Developed and ran on DEC PDP-7
  - $72,000 for minimal system
  - Main memory 9K
  - DECtape Random Access System 2.7Mbits/reel

- Influenced by early languages
  - ALGOL
  - JOSS
  - FOCAL

History 1970s

- ANSI Standard, X11.1-1977
- Veterans' Administration begins computerizing nationwide

History 1980s

- FIPS Standard, 1987
- Numerous companies implement MUMPS
  - DSM DEC
  - ISM InterSystems
  - GT.M Greystone Technology Corp
  - DTM DataTree Inc
  - MSM Micronetics Design Corporation

History 1980s

- Runs on many platforms
  - PDP-11
  - VAX/VMS
  - DEC Alpha
  - DG AViiON
  - RS/6000
  - Intel x86

History 1990s

- ISO Standard 11756-1992
- InterSystems acquires most competitors
- Vetarans’ Administration system consolidates into VistA

History 2000s

- TD Ameritrade begins using InterSystems’ Cache’ for its TPS
- Veterans’ Administration medical database largest in the world
- European Space Agency using Cache’ for its Gaia mission

History Today

- Healthcare and Financial Services
- InterSystems' Cache' and GT.M
- Several open-source implementations

System Features

The Language

B reak
C lose
D o
E lse
F or
G oto
H ang
I f
J ob
K ill
L ock
M erge
N ew
O pen
Q uit
R ead
S et
U se
V iew
W rite
X ecute

$A scii
$C har
$D ata
$E xtract
$F ind
$G et
$H orolog
$I O
$J ustify
$L ength
$NA me
$O rder
$P iece
$Q uery
$R andom
$S elect
$T ext
$TR anslate
$V iew

Structured programming language

- Supports subroutines and functions

No reserved words

Context sensitive

- Elements are defined based on context

Single universal datatype

- Strings can be coerced into Integers, Floating Point, Booleans

Arrays

- Sparse
- Associative

Postconditionals

- Control execution of commands

Built-in RegEx-like expressions

No operator precedence

- Left-associative

The Language

B reak
C lose
D o
E lse
F or
G oto
H ang
I f
J ob
K ill
L ock
M erge
N ew
O pen
Q uit
R ead
S et
U se
V iew
W rite
X ecute

$A scii
$C har
$D ata
$E xtract
$F ind
$G et
$H orolog
$I O
$J ustify
$L ength
$NA me
$O rder
$P iece
$Q uery
$R andom
$S elect
$T ext
$TR anslate
$V iew

Lazy evaluation

- @ - indirection operator

Self-Interpreted, dynamic linking

- Using eXecute command

First-Class functions

- Can be coerced through indirection

I/O handled through Devices

- O and U define the device

Built-In persistence

- ^ - global operator

Vendor specific

- Z commands
- I/O parameters for O and U
- V and $V

The Database

Hierarchical database

- Persistent
- ACID (Atomicity, Consistency, Isolation, Durability)
- Schema-less
- Key-value pairs
- B-trees

Vendor specific

- Length of global name
- Size of global data blocks
- Number of keys in a global
- Total length of global reference

The Environment

Multi-User

- Multiple users can run routines and access globals simultaneously

Multi-Tasking

- Processes can spawn other processes

Distributed Databases/Processes

- Create remote jobs
- Access remote databases
- Network locking
- Syntax is somewhat standardized
- Implementation is up to vendor

Vendor specific

- Devices - File I/O, TCP/IP, Pipes, Shared Memory, Ports
- Database management
- Process management
- Lock management
- Database mapping
- Journaling

Example

Sieve of Eratosthenes

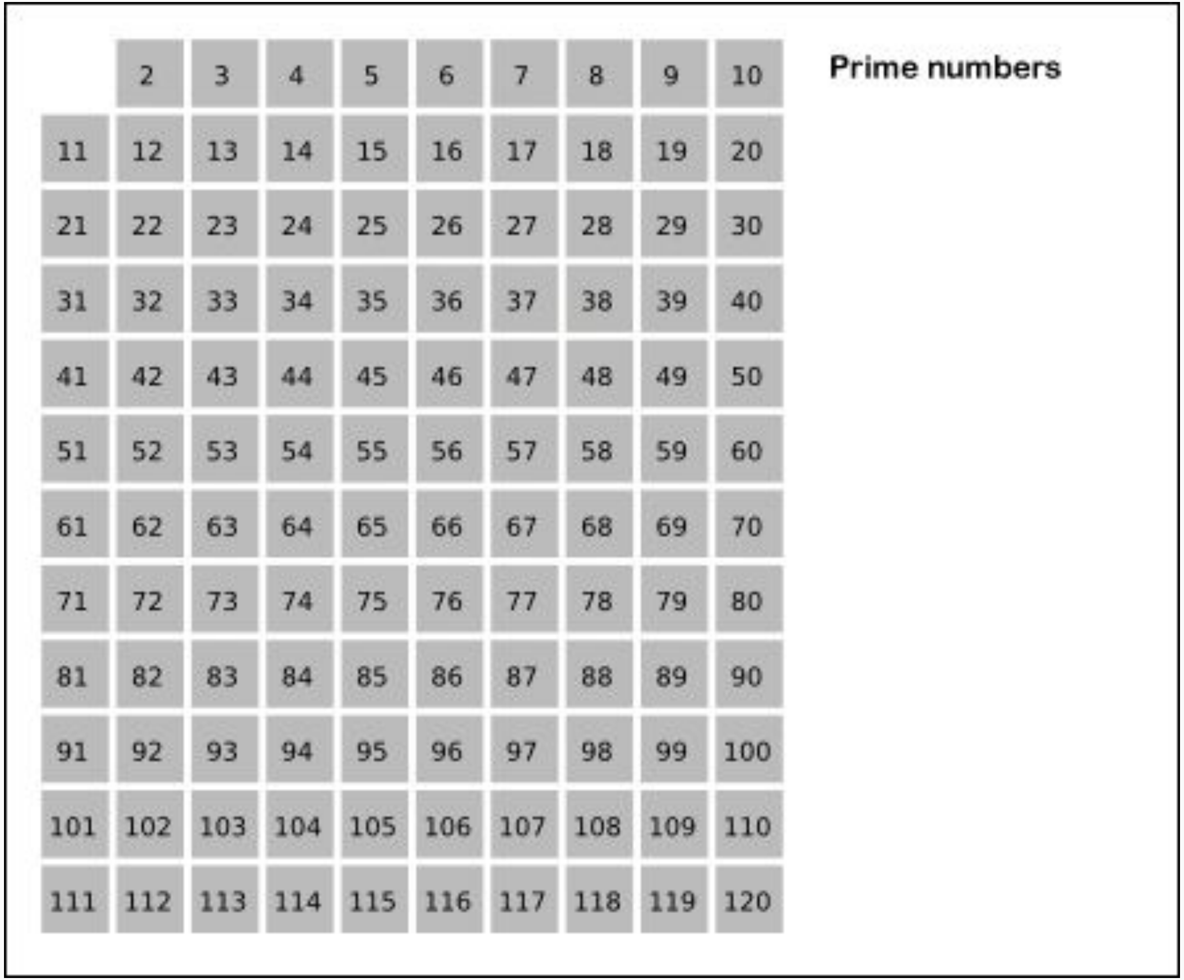
To find all of the prime numbers less than or equal to a given integer n

1. Create a list of consecutive integers from 2 through n: (2, 3, 4, ..., n).
2. Initially, let p equal 2, the smallest prime number.
3. Enumerate the multiples of p by counting to n from 2p in increments of p, and mark them in the list (these will be 2p, 3p, 4p, ...; the p itself should not be marked).
4. Find the first number greater than p in the list that is not marked. If there was no such number, stop. Otherwise, let p now equal this new number (which is the next prime), and repeat from step 3.
5. When the algorithm terminates, the numbers remaining not marked in the list are all the primes below n.

Wikipedia

Sieve of Eratosthenes

To find all of the prime numbers less than or equal to a given integer n



Wikipedia

Sieve of Eratosthenes

To find all of the prime numbers less than or equal to a given integer n

```
F I=2:1:120 S A(I)=1
F P=2:1:120 I A(P)=1 F Q=2*P:P:120 S A(Q)=0
F I=2:1:120 I A(I)=1 W I," "

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61
67 71 73 79 83 89 97 101 103 107 109 113
```

```
F I=2:1:120 S A(I)=1
S P="" F  S P=$O(A(P)) Q:P=""  W:$D(A(P)) P," " F
Q=2*P:P:120 K A(Q)

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61
67 71 73 79 83 89 97 101 103 107 109 113
```

```
F I=2:1:120 S A(I)=1
S P="" F  S P=$O(A(P)) Q:P=""  S Q=P F  S
Q=$O(A(Q)) W:Q="" P," " Q:Q=""  I '(Q#P) K A(Q)

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61
67 71 73 79 83 89 97 101 103 107 109 113
```

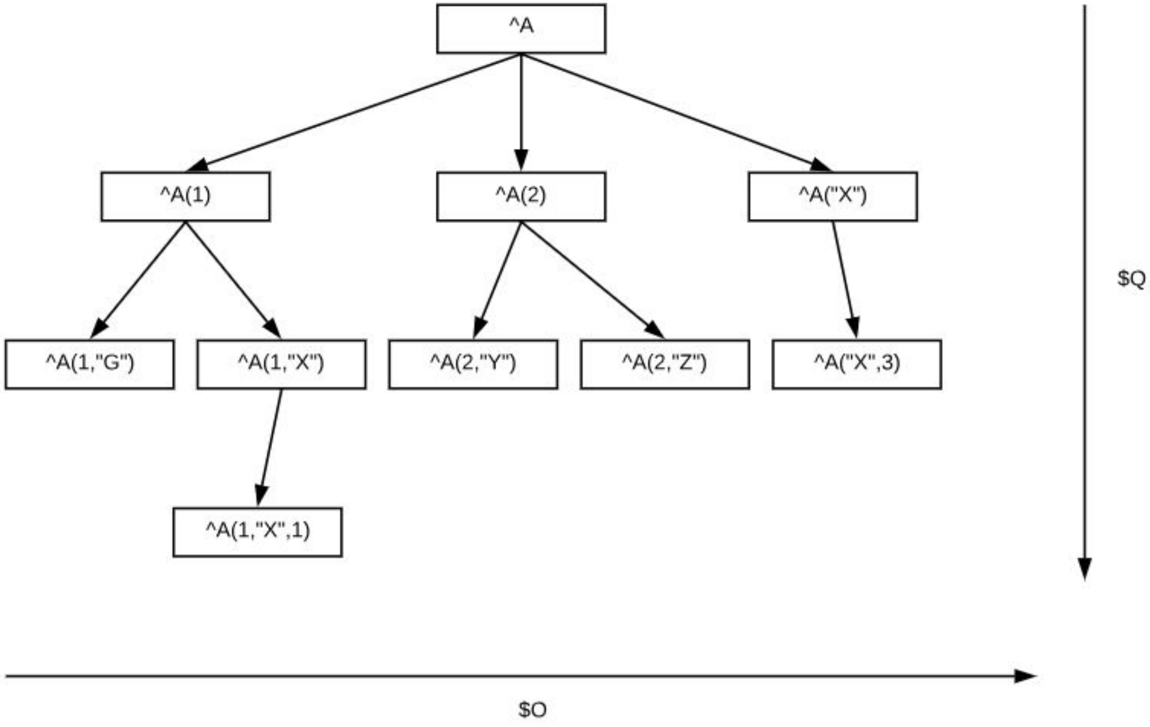
Sieve of Eratosthenes

To find all of the prime numbers less than or equal to a given integer n

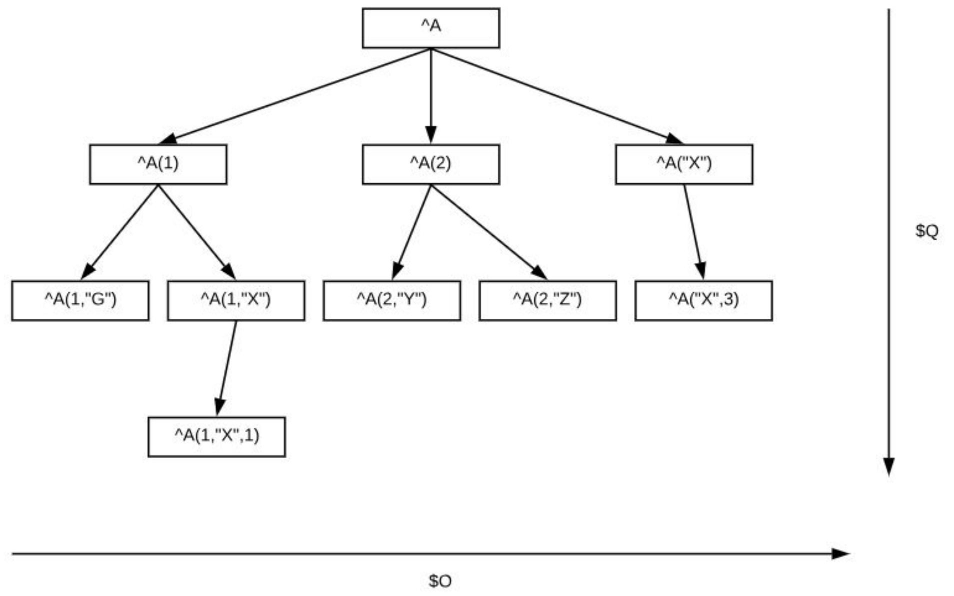
```
F I=2:1:120 S ^A(I)=1
F P=2:1:120 I ^A(P)=1 F Q=2*P:P:120 S ^A(Q)=0
F I=2:1:120 I ^A(I)=1 W I," "
```

```
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61
67 71 73 79 83 89 97 101 103 107 109 113
```

Climbing the Tree



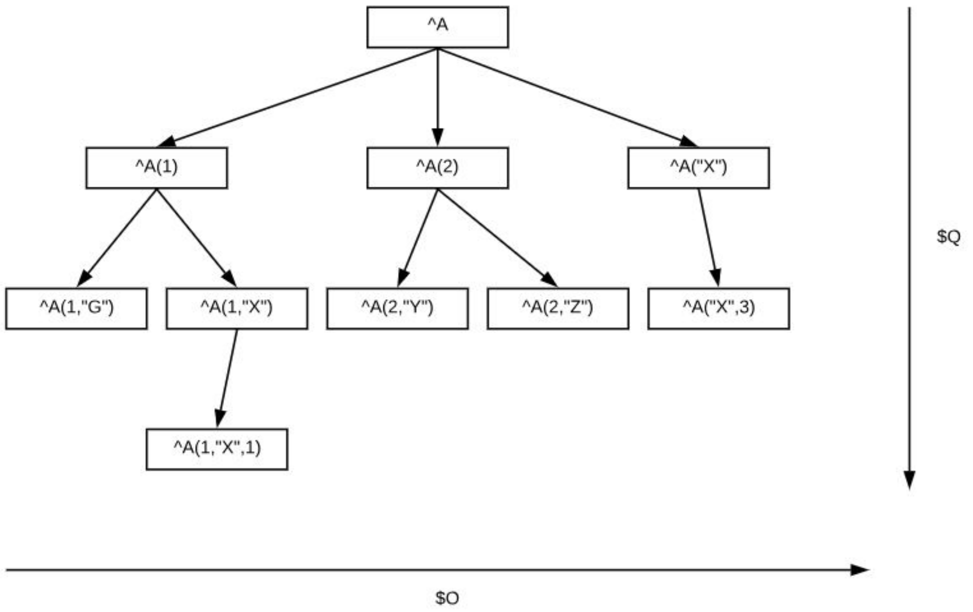
Climbing the Tree




```
S X="A" F  S X=$Q(@X) Q:X=""  W X," "
```

```
A(1) A(1,"G") A(1,"X") A(1,"X",1) A(2) A(2,"Y")
A(2,"Z") A("X") A("X",3)
```

Climbing the Tree




```
S X="" F  S X=$O(A(X)) Q:X=""  W X," "

1 2 X
```

```
S X="" F  S X=$O(A(1,X)) Q:X=""  W X," "

G X
```

```
S X="" F  S X=$O(A(2,X)) Q:X=""  W X," "

Y Z
```

```
S X="" F  S X=$O(A("X",X)) Q:X=""  W X," "

3
```

MUMPS Sorting

Sort the numbers from 0 - 9

```
F I=1:1:10 R "ENTER A NUMBER: ",X,! S ^A(X)="" I I=10 S X=""
F  S X=$O(^A(X)) Q:X=""  W X," "
```

```
ENTER A NUMBER: 5
ENTER A NUMBER: 8
ENTER A NUMBER: 2
ENTER A NUMBER: 0
ENTER A NUMBER: 1
ENTER A NUMBER: 3
ENTER A NUMBER: 7
ENTER A NUMBER: 4
ENTER A NUMBER: 6
ENTER A NUMBER: 9

0 1 2 3 4 5 6 7 8 9
```

MUMPS Sorting

B-Trees

- Self-balancing
- Keeps keys in sorted order
- All leaves are at the same depth
- Efficient for disk I/O

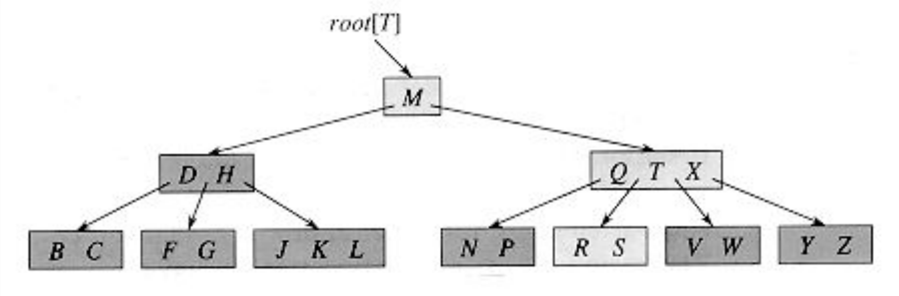


Image from Introduction to Algorithms by Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest

A Little Indirection

```
F I=1:1:10 R "ENTER A NUMBER: ",X,! S A(X)="" I I=10 S X="" F
S X=$O(A(X)) Q:X=""  W X," "
```

```
S V="A"
F I=1:1:10 R "ENTER A NUMBER: ",X,! S @V@(X)="" I I=10 S X=""
F  S X=$O(@V@(X)) Q:X=""  W X," "
```

```
ENTER A NUMBER: 7
ENTER A NUMBER: 2
ENTER A NUMBER: 4
ENTER A NUMBER: 3
ENTER A NUMBER: 8
ENTER A NUMBER: 1
ENTER A NUMBER: 5
ENTER A NUMBER: 6
ENTER A NUMBER: 9
ENTER A NUMBER: 0

0 1 2 3 4 5 6 7 8 9
```

A Little Indirection

```
S V="A"
F I=1:1:10 R "ENTER A NUMBER: ",X,! S @V@(X)="" I I=10 S X=""
F  S X=$O(@V@(X)) Q:X=""  W X," "
```

```
S V="A",N=10,S="ENTER A NUMBER: "
F I=1:1:@N W S R X,! S @V@(X)="" I I=@N S X="" F  S
X=$O(@V@(X)) Q:X=""  W X," "
```

```
ENTER A NUMBER: 9
ENTER A NUMBER: 8
ENTER A NUMBER: 7
ENTER A NUMBER: 6
ENTER A NUMBER: 5
ENTER A NUMBER: 4
ENTER A NUMBER: 3
ENTER A NUMBER: 2
ENTER A NUMBER: 1
ENTER A NUMBER: 0

0 1 2 3 4 5 6 7 8 9
```

A Little Indirection

```
S V="A",N=10,S="ENTER A NUMBER: "
F I=1:1:@N W S R X,! S @V@(X)="" I I=@N S X="" F  S
X=$O(@V@(X)) Q:X=""  W X," "
```

```
S V="A",N=10,S="ENTER A NUMBER: "
S C="S X="""" F  S X=$O(@V@(X)) Q:X=""""  W X,"" """
F I=1:1:@N W S R X,! S @V@(X)="" I I=@N X C
```

```
ENTER A NUMBER: 3
ENTER A NUMBER: 6
ENTER A NUMBER: 8
ENTER A NUMBER: 2
ENTER A NUMBER: 1
ENTER A NUMBER: 0
ENTER A NUMBER: 9
ENTER A NUMBER: 7
ENTER A NUMBER: 4
ENTER A NUMBER: 5

0 1 2 3 4 5 6 7 8 9
```

A Little Indirection

```
S V="A",N=10,S="ENTER A NUMBER: "
S C="S X="""" F  S X=$O(@V@(X)) Q:X=""""  W X,"" """
F I=1:1:@N W S R X,! S @V@(X)="" I I=@N X C
```

```
S V="A",N=10,S="ENTER A NUMBER: "
S C="S X="""" F  S X=$O(@V@(X)) Q:X=""""  W X,"" """
S C2="F I=1:1:@N W S R X,! S @V@(X)="""" I I=@N X C"
X C2
```

```
ENTER A NUMBER: 3
ENTER A NUMBER: 2
ENTER A NUMBER: 6
ENTER A NUMBER: 9
ENTER A NUMBER: 0
ENTER A NUMBER: 1
ENTER A NUMBER: 4
ENTER A NUMBER: 5
ENTER A NUMBER: 7
ENTER A NUMBER: 8

0 1 2 3 4 5 6 7 8 9
```

Fin.