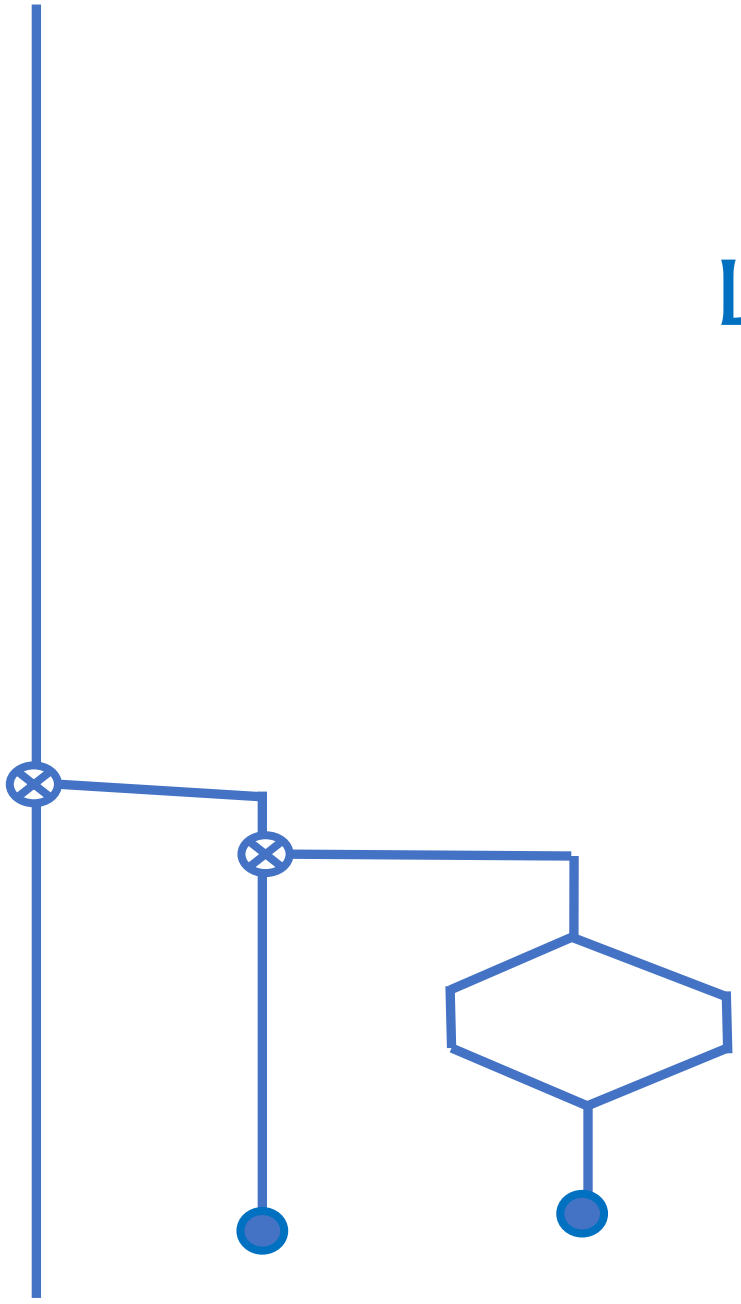


Levels Of Abstraction Using OOP

SVFIG
Jan. 24, 2026
Bill Ragsdale

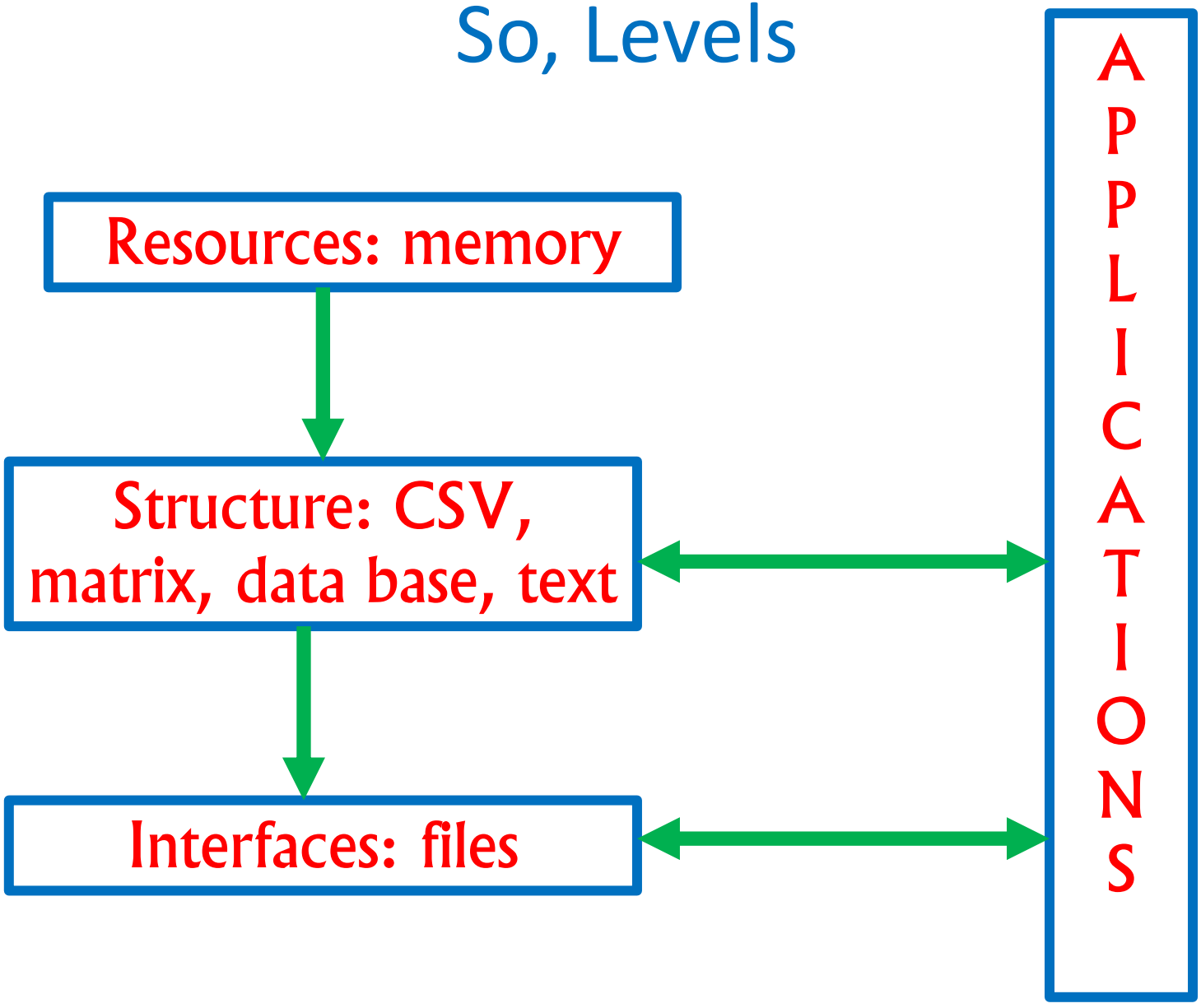
The Rationale

- I'm using a variety of OOP objects.
  - Matrix
  - Data base
  - CSV
- I wanted to bring a commonality to them.
- And match them to the logical flow of my application's development.

So, Levels

- OOP allows objects to 'inherit' the data structure and methods of prior objects.
- We'll use three levels.

So, Levels


Resources: memory
Structure: CSV, matrix, data base, text
Interfaces: files
APPLICATIONS

Levels, and

- Memory allocator is the same for all.
- Data objects specific for text, CSV, data-base, matrices.
- File interface, same for all.
- Each final object carries all the lower level parameters and methods.

Memory Allocator

```
:CLASS :Allocation/   <Super Object

      int Location  int Size

:M Release:  Location if Location release 0
to Location then ;M

:M PutLocSize: ( loc size --- ) Release:
self to Size to Location ;M

:M GetLocSize: ( --- loc size ) Location
Size ;M

;CLASS
```

Support

```
: /Initialize ( object size  --- )
    over Release: :Allocation/
    dup MALLOC swap  2dup erase
    rot PutLocSize: :Allocation/ ;
: /Dump ( object --- ) \ display memory
    GetLocSize: :Allocation/  dump ;
[
```

Allocation Example

```
:Allocation/  An-Allocation/

An-Allocation/ 64 /Initialize

An-Allocation/ /Dump
```

```
testing :Allocation/
  E4D570 | 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
  E4D580 | 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
  E4D590 | 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
  E4D5A0 | 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

A Data Base

```
:Class :DB/   <Super :Allocation/    \ data base object
   int #Records int #Fields int Record-size int Field-size
:M SetDBparams:
   to field-size to record-size to #fields to #records ;M
:M GetFieldAddress:
      Field-size * swap Record-size * + Location + ;M
:M Get#R#F:  #Records #Fields ;M
;CLASS
```

DB Support

```
: /InitDB  ( DBobject fldsize #recs #flds --- )
 {: field-size #records #fields | record-size:}
  field-size #fields * dup to record-size
  #records * to size
  dup size /Initialize
  #records #fields record-size field-size
   4 roll SetDBparams: :DB/ ;
```

DB Support, more

```
: /LoadDB
   dup Get#R#F: :DB/
   swap 0 do  dup  0 do
  j i  3 pick GetFieldAddress: :DB/
  j 100 * i + swap ! loop cr loop 2drop ;

: /ListDB
   dup 2 cells- >name count 3 spaces type cr
   dup Get#R#F: :DB/
   swap 0 do  dup  0 do
     j i  3 pick GetFieldAddress: :DB/
     @  6 .r loop cr loop 2drop ;
```

Testing DB

```
:DB/  A-DB/  \ create db object.

A-DB/ 4 3 4 /InitDB \ size rec fld

A-DB/       /LoadDB

A-DB/       /ListDB

Release: A-DB/  Forget A-DB/
```

```
A-DB/
  0    1    2    3
100  101  102  103
200  201  202  203
 ok
```

File Read-Write

```
:Class :DBfile/  <Super :DB/
      maxstring bytes Path
:M PutPath:   count Path place ;M
:M PathToPad:  Path count pad place
      self 2 cells- >name count 1- pad +place
                    c" .db"  count pad +place ;M
:M ReadFile:  PathToPad: self pad count
    r/o open-file -280 ?throw
   dup>r *file-size -281 ?throw
   dup MALLOC  swap 2dup r@ read-file
   -281 ?throw  drop  handle
   r> close-file  abort" File closing error"
   PutLocSize: self  ;M
:M WriteFile:  GetLocSize: self
     PathToPad: self pad  Fsave-file  ;M  ;CLASS
```

File Read-Write, support

```
: /WriteDBfile WriteFile: :DBfile/  ;

: /ReadDBfile  ReadFile: :DBfile/ ;

: /ReleaseFile Release: :DBfile/  ;

: /PutPath      swap PutPath: :DBfile/ ;
```

Example

```
:DBfile/  SUGTX/  \ create the data base file
SUGTX/ 4 3 4 /InitDB \ fieldsize, record size
                     \ fields/record
SUGTX/  /LoadDB
SUGTX/  /ListDB cr
```

```
SUGTX/
  0     1     2     3
100   101   102   103
200   201   202   203
ok
```

Example, continued

```
SUGTX/ c" C:\Data\Forth\Application
           Tools\Object Master\" /PutPath
SUGTX/  /WriteDBfile
SUGTX/  /ReadDBfile
SUGTX/  /ListDB
SUGTX/  /ReleaseFile
Forget SUGTX/
```

```
SUGTX/
  0     1     2     3
100   101   102   103
200   201   202   203
ok
```

Summary

Originally, I was rewriting the allocation, the structure and read/write for each application.

About 150 lines of Forth code is application independent.

The middle layer, ‘structure’ is unique for each application. CSV, DB, matrix, text etc.