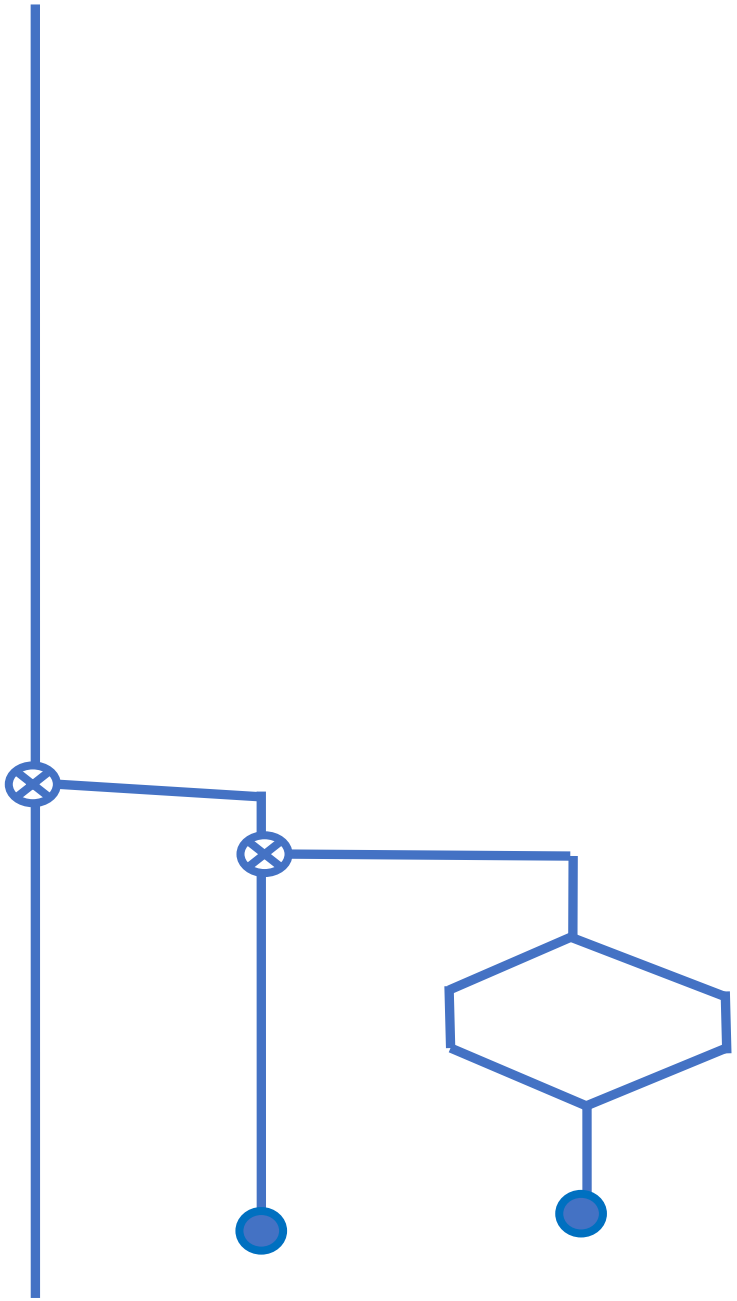

File Access Using O.O.P.

SVFIG

July 26, 2025

Bill Ragsdale

Today

I have an extensive history of financial security files.

The format has been modified by several vendors over the last forty-years.

It uses a mix of binary integers, Microsoft Qbasic Floats and IEEE32 floats. I use IEEE64 bit floats.

I'm programming analysis software in Win32Forth.

I've expanded into using Object Oriented Programming.

Today

We'll summarize object-oriented programming.

Then see the summary report generator.

The summary report.

The security weekly report generator.

A security report.

The Basics

An object is the ‘work-horse’. It contains memory allocations and methods.

Many similar objects may be defined in a Class.

All access to object memory must go through Methods.

Methods are defined by Forth in the usual manner.

Methods are common to all objects of a class.

But, each object has its own memory allocations.

Class Creation, of object definer

```
:CLASS <object-definer>   <Super Object
      <memory allocations>
   :M  <method:>  <code> ;M
    .
    .
   :M  <method:>  <code> ;M
;CLASS
```

My Object Definer Filer/

```
:CLASS Filer/  <Super Object
  200 bytes Path    20 bytes Name
      int Handle    int Size
      int Location \ allocation in memory
      int Offset   \ of records in the flle
```

Methods

```
:M Set-name:  ( addr count --- )   Name place  ;M
:M Form-path: ( addr count  --- )
     Path place  Name count Path +place ;M
:M Open: \ loads File-handle
     Path count r/o open-file      \  file-id ios
    -280 ?throw  to Handle  ;M     \ file-id into EM-handle
:M Size:  Handle sp@ 0 swap rot
     call GetFileSize              \ 0 size  or  -1 -1
     nip dup INVALID_FILE_SIZE = abort" size error"
     to Size  ;M  \ size
:M Allocate: \ creates   storage-address
    Location if Location release then \ in case it is open
    Size MALLOC to Location ;M
:M Close:  Handle close-file  abort" File closing error"  ;M
```

Methods

```
 :M Read:  ( --- file-location file-size )
     Open: self   Size: self  Allocate: self \ get memory
     Location Size  Handle  read-file
     281 ?throw  drop  Close: self   ;M
:M Set-offset:  to Offset ;M
:M >addr: ( fileID offset --- addr )
    swap offset * Location + + ;M
:M Release:  Location \ adjust for EMASTER or DAT file
    if  Offset 50 >
        if s" EMASTER release" else s" DAT release" then   type cr
       Location release  0 to Location \ Return memory allocation.
     then ;M
:M ClassInit: 0 to Location ;M
```

Comments

The Methods, above could be combined and simplified.

However, I factored them for testing.

EMASTER is the index for all securities files.

DAT is a file of one security over time.

Creating The EMASTER Object

Filer/ EMASTER

We now have the storage and methods specific to EMASTER.

Opening EMASTER

```
: Open-EMASTER
                192 Set-offset: EMASTER   \ bytes/record
   s" EMASTER"        Set-name: EMASTER
   Proto-path count  Form-path: EMASTER
                         Read: EMASTER  ;
Open-EMASTER

I'll skip the word definitions to access fields within
       EMASTER.
```

EMASTER Report Generator

```
: EM-line ( fileID --- ) \ one line of EMASTER report
  dup                3 .r                    6 spaces
  dup DAT-file#      2 .r                    4 spaces
  dup DAT-symbol     tidy-symbol 6 type+     2 spaces
  dup DAT-name       21 type+                2 spaces
  dup DAT-first-dt   date>pad 10 type+       2 spaces
      DAT-last-dt    date>pad type  ;

: EM-report ( --- ) \ multiline over fileIDs
   cr cr EM-header count type cr
   Files-present 1+  1  do i EM-line cr loop ;
```

EMASTER Report

ID	File#	Symbol	Name	First dt	End dt
1	1	FBGRX	Fid Blue Chip Growth	1/6/1989	9/29/2023
2	2	FDRXX	Fid Govt CR	12/1/1989	11/19/2021
3	3	FGOVX	Fid Govt Income	4/3/1998	9/29/2023
4	5	FBNDX	Fid Invest Grade	7/7/1989	9/29/2023
5	6	FMSFX	Fid Mort Sec	1/6/1995	9/29/2023
6	7	FOCPX	Fid OTC	7/3/1986	9/29/2023
7	8	FOSFX	Fid Overseas	1/6/1989	9/29/2023
8	9	FPBFX	Fid Pac Basin	12/31/1992	9/29/2023
9	11	FTRNX	Fid Trend	1/3/1986	9/29/2023
10	12	FDVLX	Fid Value	1/6/1989	9/29/2023

Declaring The DAT File

```
 Filer/ MS-Data     \ Create Data file object

20 Set-offset:  MS-Data  \ Insert record size

: >dat ( week# offset --- addr )
 \ data request to its address
        >addr: MS-Data ;
```

Reading A DAT File

```
: DAT-load ( fileID ---  ) \ one security file
  1 max   Files-present min
  dup to DAT-loaded    \ opening a DAT file
  DAT-file#            \ translate to F#.DAT
  pad off  s" F" pad place  n>pad \ file number
        s" .dat" pad +place
  pad count          Set-name: MS-Data
  Proto-path count  Form-path: MS-Data
                        Read: MS-Data ;
```

DAT Records Access

```
: DAT-records ( --- n ) 0 2 >dat W@ ;
: DAT-date  ( wk# --- fdate )
                0 >dat @ MS>IEEE64  19000000e f+ ;
: DAT-high  ( wk# --- f )   4 >dat @  ms>IEEE64 ;
: DAT-low   ( wk# --- f )   8 >dat @  ms>ieee64 ;
: DAT-close ( wk# --- f )  12 >dat @  ms>ieee64 ;
: DAT-vol   ( wl# --- f )  16 >dat @  ms>ieee64 ;
```

DAT Report Generator

```
: DAT-line ( week# --- )
   dup              4 .r 2 spaces
   dup  DAT-date    date>pad 10 type+
   dup  DAT-high    8 2 F.R
   dup  DAT-low     8 2 F.R
   dup  DAT-close   8 2 F.R
        DAT-vol     f>s 6 .R  ;
: DAT-report ( week#  --- )
   DAT-info  DAT-header count type cr
   20 over + DAT-records  min swap
       do i DAT-line cr loop ;
```

DAT Report

Fid Blue Chip Growth FBGRX

FileID 1 F1.DAT 1/6/1989 9/29/2023

Rec	Date	High	Low	Close	Vol
100	11/30/1990	14.50	13.79	14.06	0
101	12/7/1990	14.85	14.14	14.21	4000
102	12/14/1990	14.76	14.16	14.20	0
103	12/21/1990	14.78	14.14	14.34	0
104	12/28/1990	14.77	14.27	14.29	0
105	1/4/1991	14.87	13.89	13.89	0
106	1/11/1991	14.20	13.56	13.77	0
107	1/18/1991	15.12	13.67	14.67	0
108	1/25/1991	15.52	14.56	15.05	0

Summary

- Handling similar files easily led to using object oriented programming.
- Occasional research led to an inventory of floating-point conversion routines.
- Next is to build a catalog of analysis methods. Also, likely for OOP.
- As for so much progamming, floating point conversions were difficult at first and then easy.