Published: 03/08/67

Identification

Varying String Initialize and Cleanup Routine
Varst_
S. H. Webber

Purpose

The routine varst_ is called once for each automatic data aggregate containing varying strings upon entering a block and once for each static aggregate containing varying strings upon first reference. It is called again (at another entry point) for each such automatic data aggregate upon leaving the block. The initialize entry, varst_$zero, sets the lengths of all varying strings in the data aggregate to zero. The cleanup entry point, varst_$clear, frees all varying strings in the data aggregate which have non-zero lengths.

Implementation

Upon leaving a block of procedure code any varying strings which have been used in the block still exist in free_. These must be freed in order to prevent storage from being irretrievably lost. However, in order to free the storage used by such varying strings, some criterion must be established which allows a test to see whether the string must be freed or not, (usually whether the string was used or not.) The length of the string satisfies this requirement - if the length is zero, the string need not (and should not) be freed. In order to implement this method, all automatic varying strings must have their lengths set to zero upon entering a block. This is the task of varst_$zero. Then upon leaving a block only those varying strings which have non-zero lengths need be freed. This is the task of varst_$clear. The freeing is done by calls to freen_, one call for each varying string found. For structures which have non-elementary substructures varst_ calls itself recursively.

Varst_ is called by either:

```
call varst_$zero (specifier)
```

or

```
call varst_$clear (specifier)
```

where specifier is the specifier of the data aggregate containing varying strings.

The method used by varst_ is described in Figure 1. (Note: most of the work done by varst_ is in deciphering the dope of the data aggregate. Since this must be done for both entry points, only limited parts of the program distinguish between the two entry points. The distinction is established in the initializing stages for each entry.)

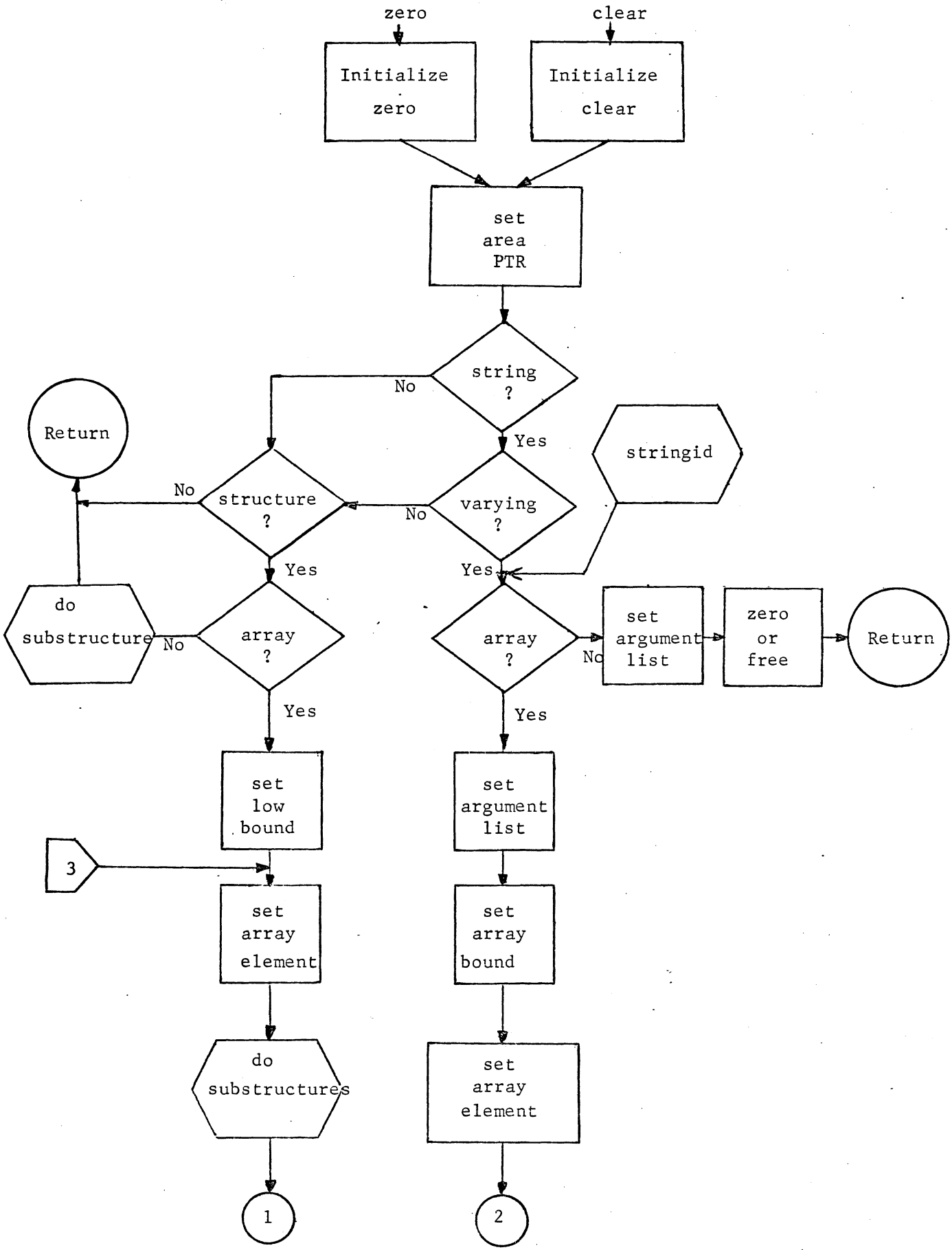

zero
clear
Initialize zero
Initialize clear
set area PTR
string ?
No
Yes
Return
stringid
structure ?
No
varying ?
No
Yes
Yes
do substructure
No
array ?
array ?
No
set argument list
zero or free
Return
Yes
Yes
set low bound
set argument list
3
set array element
set array bound
do substructures
set array element
1
2

Figure 1

Figure 2

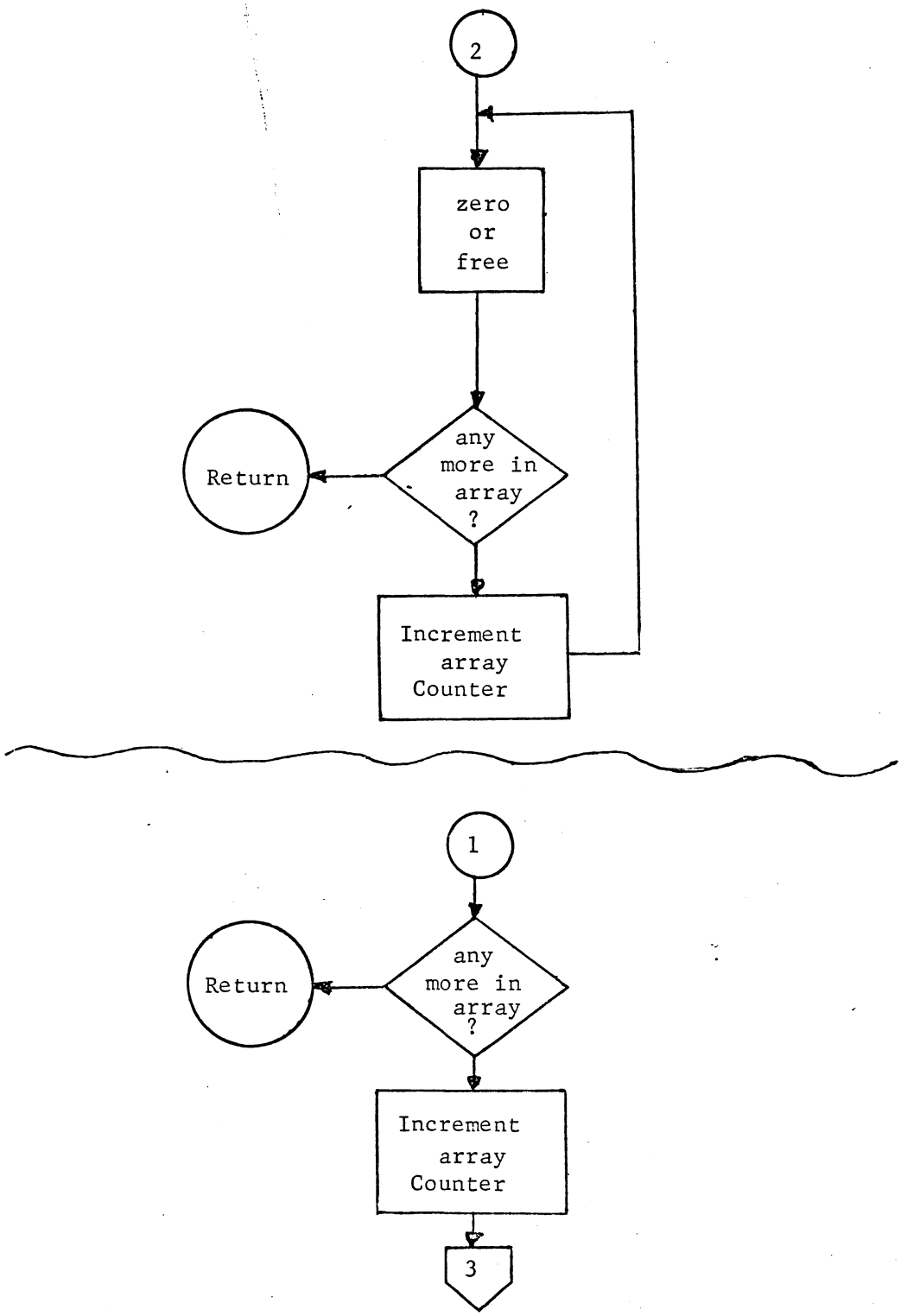
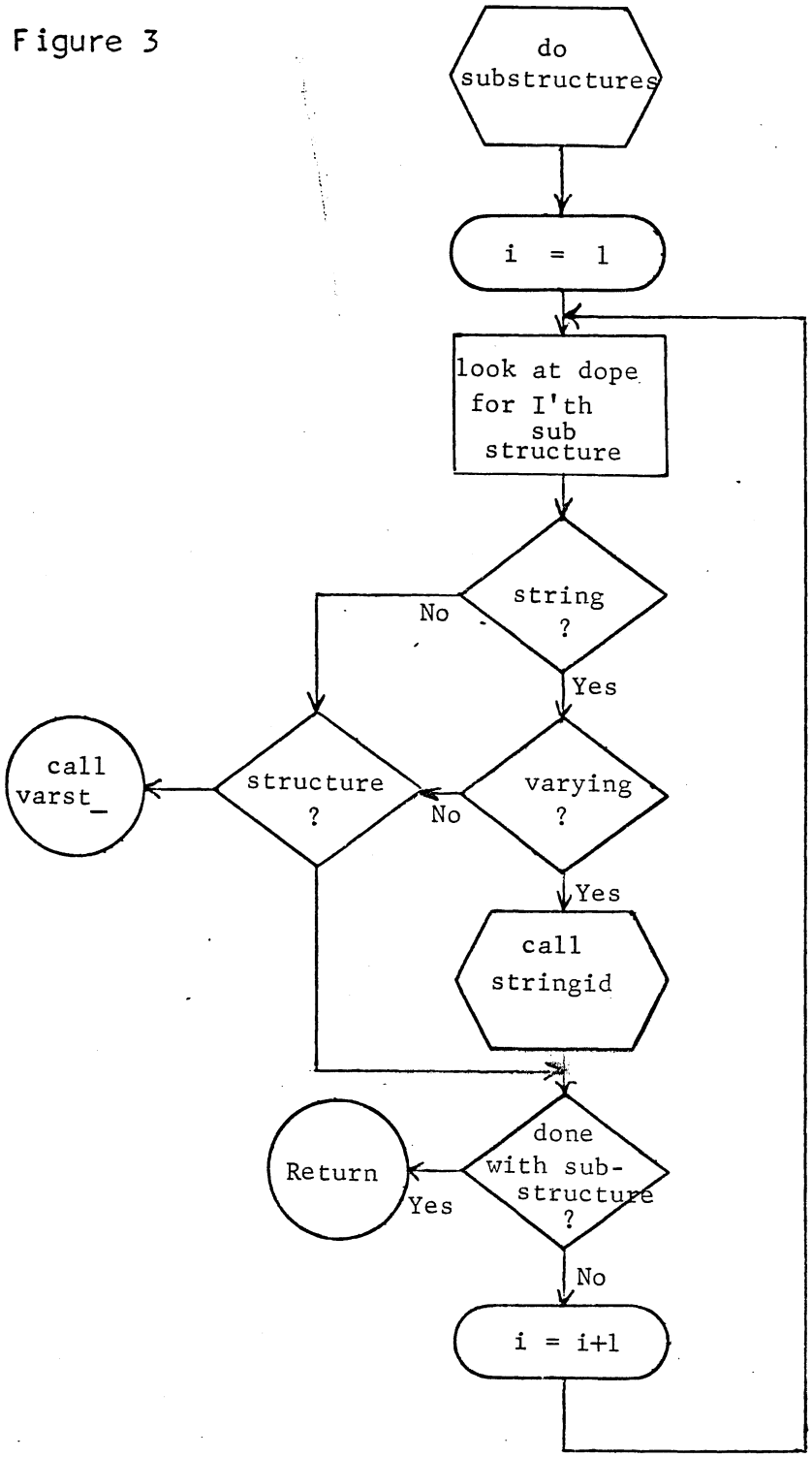

2
zero or free
any more in array ?
Return
Increment array Counter
1
any more in array ?
Return
Increment array Counter
3

Figure 3


do substructures
i = 1
look at dope for I'th sub structure
string ?
No
Yes
varying ?
No
Yes
structure ?
call varst_
call stringid
done with sub-structure ?
Yes
Return
No
i = i+1