Manual Memory Management

In computer science, manual memory management refers to the usage of manual instructions by the programmer to identify and deallocate unused objects, or garbage.

----------WIKI

garbage collection

In computer science, garbage collection (GC) is a form of automatic memory management. The garbage collector, or just collector, attempts to reclaim garbage, or memory occupied by objects that are no longer in use by the program.

reference counting

As a collection algorithm, reference counting tracks, for each object, a count of the number of references to it held by other objects. If an object's reference count reaches zero, the object has become inaccessible, and can be destroyed.

Mark and sweep

Mark and sweep is a technique in Garbage Collection to free all unreferenced objects.

The first stage is the mark stage which does a tree traversal of the entire 'root set' and marks each object that is pointed to by a root as being 'in-use'. All objects that those objects point to, and so on, are marked as well, so that every object that is reachable via the root set is marked.

In the second stage, the sweep stage, all memory is scanned from start to finish, examining all free or used blocks; those not marked as being 'in-use' are not reachable by any roots, and their memory is freed. For objects which were marked in-use, the in-use flag is cleared, preparing for the next cycle.

Explain VM design

Objects

Any object that’s being referenced by a variable that’s still in scope is in use.

Any object that’s referenced by another object that’s in use is in use.

two types of objects: ints and pairs

A pair can be a pair of anything, two ints, an int and another pair, whatever.

 virtual machine

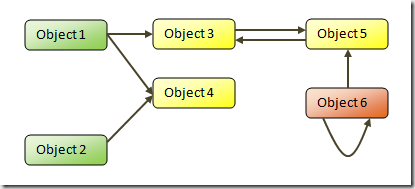
Its role in this story is to have a stack that stores the variables that are currently in scope.

Stack

there is actually still a stack. It’s used to store local variables and temporary variables needed in the middle of an expression.

The main Object struct has a type field that identifies what kind of value it is— either an int or a pair. Then it has a union to hold the data for the int or pair.

*Object*: This is a unit of storage on the heap. It generally means an object in the object oriented sense but is equally applicable to a procedural language (e.g. a struct/native-type in C) or functional language.

*Object/Reference graph*: This is the directional graph of objects in memory. A typical sample is below. The nodes are objects in memory and the edges (arrows) are references one object holds to another (e.g. a pointer or reference in C/C++). There can be circular references between two nodes (Object 3 and Object 5) or nodes referencing themselves (Object 6).   
 [](https://msdnshared.blob.core.windows.net/media/TNBlogsFS/BlogFileStorage/blogs_msdn/abhinaba/WindowsLiveWriter/BackToBasicsMarkandSweepGarbageCollectio_FF09/image_4.png) 

*Roots*: These are the set of nodes in the object graph from which the references start. These are typically references held in registers, local variable on the stack or global variables. The green nodes in the diagram above are roots.

*Unreachable object*: These are nodes in the graph which have no edge referencing them. The Orange node in the diagram above is an unreachable object. This is the node that the GC needs to clean/free because it is not reachable from any node and is hence garbage memory.

Allocation

Most languages we deal with today support a whole bunch of different memory allocation options. We have an option to use static allocation, stack-allocation and heap allocation.