Overview

The term *garbage collection* refers to the process of finding heap variables that have become inaccessible, and deallocating the memory storing the data. IDL performs automatic garbage collection on all data types so that when a variable is destroyed or goes out of scope, the variable's memory is automatically reclaimed.

IDL garbage collection is implemented using a reference counting scheme. In reference counting, each heap variable keeps track of the number of references to it. In general, the reference count increases when a new reference to the underlying heap variable is created, and decreases when a reference is removed. When the reference count reaches zero, the object or pointer is destroyed. You can call the `HEAP_REFCOUNT` function to return the reference count for a pointer or object reference. You can also call the `HELP` procedure with the `/FULL` keyword to retrieve the same information.

An object or heap variable can have multiple references to it. When the last reference to an object is removed, the object's `Cleanup` method is called and the heap variable is automatically destroyed. When the last pointer to a heap variable is removed, the associated heap variable is automatically destroyed.

Note: Automatic garbage collection is unable to destroy the objects in a cyclic reference (where two objects or pointers refer to each other, but no other object or pointer refers to either).

In addition to automatic garbage collection, users have several options for explicitly deallocating heap variables. Users can call `PTR_FREE` or `OBJ_DESTROY` to explicitly free associated pointers and objects. A user can also call `HEAP_FREE` to recursively free all heap variables that are contained within an input variable. Finally, the `HEAP_GC` function can be called to perform garbage collection on all inaccessible heap variables.

Enabling and Disabling Automatic Garbage Collection

By default, automatic garbage collection is enabled for all object and pointer heap variables, except for `IDLgrWindow` and all subclasses. (When the actual window is closed, the `IDLgrWindow` is automatically destroyed.)

You can disable automatic garbage collection for all heap variables using the `HEAP_REFCOUNT` function. This is useful when testing an application to ensure that variable cleanup is working as expected.

If you want to disable automatic garbage collection for a particular object reference or pointer, call `HEAP_REFCOUNT` with the pointer or object reference and the `DISABLE` keyword.

If you want to always disable garbage collection for a specific object class, you can add the following line of code to the class `Init` method:

```
void=HEAP_REFCOUNT(self, /DISABLE)
```

This code will disable garbage collection for all class instances.

To determine if garbage collection is enabled for a particular heap variable, call `HEAP_REFCOUNT` with the `IS_ENABLED` keyword.