Base Language - Bug #12117

automatically clear memptr addresses

11/04/2026 03:37 PM - Constantin Asofiei

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**reviewer:**	
vendor_id:	GCD	**production:**	No
case_num:		**env_name:**	
version_reported:		**topics:**	
version_resolved:			

Description

Related issues:

Related to Runtime Infrastructure - Feature #9684: migrate from libffi (and C...	**New**

History

#1 - 11/04/2026 03:40 PM - Constantin Asofiei

In FWD, we can track all allocated addresses and clean them up even if the 4GL code does not - this will be at the address level, with a reference counter similar as we do for OO:

- we need to track the address across assignments, parameters, etc
- internal usage of memptr - needs to be analyzed, I'm not sure if we need to create 'memptrvar' or not (similar to ObjectVar)
- get/set-pointer-value or passing the memory address to a native call - that address will be marked as 'dirty' and will not be cleaned up automatically in FWD
- otherwise, we need to track the scope of the memptr var (i.e. if the program or top-level block defining it is deleted/finished, and the reference counter for the address is zero, then we automatically release that memory)

#3 - 11/04/2026 11:54 PM - Greg Shah

To be clear: we will track reference counts to addresses not to memptr instances.

Another point: it is possible to calculate the address of an arbitrary index inside an allocated block of memory. This is something done all the time by native C code (and similar). I point this out because it is a potential anti-pattern where a returned address can be turned into a memptr or otherwise passed around (e.g. to a native API call) but it is from inside the block and might effectively be still in use after the memptr itself goes out of scope.

#4 - 11/05/2026 01:38 AM - Constantin Asofiei

Yes, any address incoming from or outgoing to native code will be 'dirty' and not automatically cleared.

#5 - 11/05/2026 01:44 AM - Greg Shah

The problem here is that it is possible for these addresses to be passed around as 64-bit integers and not necessarily as memptrs. So it is not 100% possible to catch all scenarios. But as I said, that is an anti-pattern.

#6 - 11/05/2026 01:46 AM - Constantin Asofiei

Greg Shah wrote:

> The problem here is that it is possible for these addresses to be passed around as 64-bit integers and not necessarily as memptrs. So it is not 100% possible to catch all scenarios. But as I said, that is an anti-pattern.

Don't you need get-pointer-value to get this address from an allocation via memptr? In such case, again will not be tracked.

#7 - 11/05/2026 02:53 AM - Teodor Gorghe

- Related to Feature #9684: migrate from libffi (and C heap functions) to the built-in JDK FFM support added

#8 - 11/13/2026 10:52 AM - Constantin Asofiei

Teodor, if you want to work on this, go ahead, please.

#9 - 11/13/2026 10:53 AM - Teodor Gorghe

I will handle this after finishing #12159.