Design Documentation

for

Simple reference counter

# Summary

We have implemented a simple reference counter in C, similar to the one on Java, for a automatic and smother memory management in you program.

# Objects and Metadata

To be able to use a reference counter you need objects with metadata, since C does not have object we define object as a void pointer.

To have metadata in C our solution is to when we allocate something on the heap, we allocate it together with a struct of metadata containing the object's size, reference counter and a destructor function (see “Garbage and destruct\_list”).

# Release and retain

Release and retain are simply the decreasing and increasing of a object's reference counter.

If an object’s reference counter hits zero, the object is automatically freed and removed from memory.

# Cascade limit

Usually, an object is removed as soon as it’s reference counter hits zero, but so to not halt the entire program if a very large number of objects wher to be deallocated, e.g. a long link list, we implemented a cascade limit to restrict how many object’s that can be deallocated at once.

# Deallocate before allocate

But to make sure that every object gets deleted, we do a deallocate on the remaining objects every time a new one is allocated until we either hit the cascade limit again.

But if the amount of memory freed is less than the bytes requested for allocation, we ignore the cascade limit and continue to free objects until the requested number of bytes have been freed or there are no more objects to free.

# Garbage and destruct\_list

In order to solve the problem with garbage remaining after a deallocate, we put all garbage objects (objects with a rc of zero) in to a list, so that when we do deallocate it we keep deallocating garbage from the list until we hit the cascade limit or the list is empty

To decrease the size of our metadata struct, instead of having the object’s destructor function be part of it’s metadata, we put it in a list, and just save the list index (represented by a byte) in the metadata.

# Overview

In short, to have a referens counter we represent all objects with a void pointer, and when we allocate them on the heap we put a metadata structure just before it.

So to not spend too much time at once deallocating objects we have a cascade limit and the remaining objects are saved in a list and deallocated at the next allocation.