Static Variables

A static variable is a variable allocated statically. Its extent extends across the entire run of the program.

Life Cycle of Variables

Static Variables continue to exist until the class in which they are used is loaded. Once the class is unloaded the variable ceases to exist and become part of the garbage collection. Similarly, class variables are stored in the memory until the class is executed. Once the execution is over the memory allocated to the class variable is de-allocated. The lifetime of a dynamic object begins when memory is allocated for the object (e.g., by a call to malloc()) and ends when memory is de-allocated (e.g., by a call to free()). Dynamic objects are stored in "the heap". Variables initialized in methods run and are stored in the memory only for the time during which the method is executed. These variables cannot be used outside of the methods and similarly their memory allocations are deleted once the execution of the method is over.

Dynamic Memory in C

Dynamic allocation is one of the three ways of using memory provided by the C/C++ standard. To accomplish this in C the malloc function is used and the new keyword is used for C++. Both of them perform an allocation of a contiguous block of memory, malloc taking the size as parameter. This memory block can be used whenever needed during the program execution or until explicitly de-allocating it, unlike the automatic memory which is available only inside the function or block of instructions where it was declared. Allowing a program to allocate dynamic storage every time it needs more until the program stops can cause it eventually to run out of available space. To prevent this behavior C++ provides the delete operator with the job of recycling a segment of memory allocated with new.

Class

Object

Instance

Herencia

Sobrecarga

Shadowing