**C Static**

1. **Static Variables in C**
2. When used inside a function:

* Remains in memory while program is running (not destroyed even when out of funciton scope). Their value can be re-initialized.
* Static variables are allocated in data segment, not stack segment.
* Static variables can only be initialized using constant literals
* They are initialized to 0 if not explicitly initialized.
* Static variables **cannot be declared inside a structure**. The reason is C compiler requires the entire structure elements to be placed together (i.e.) memory allocation for structure members should be contiguous. It is possible to declare structure inside the function (stack segment) or allocate memory dynamically(heap segment) or it can be even global (BSS or data segment). Whatever might be the case, all structure members should reside in the same memory segment because the value for the structure element is fetched by counting the offset of the element from the beginning address of the structure. Separating out one member alone to a data segment defeats the purpose of static variables
* *It’s possible to create a static structure in C*

1. When used at global scope 🡪 Limits the scope of the variable to the current source file)
2. **Static Functions in C**

In C, functions are global. **static** functions in C restrict access to the file only (recall that we cannot declare functions inside structs in C)

**NOTE that this static has a different meaning in C++**

**C++ Static**

1. **Static Variables (inside a function)**

Retaining values between multiple invocations of the function, just like in C

1. **Static Data Members (inside a class or a struct)**

* Only one copy of that member is created for the entire class, and is shared by all objects of that class, no matter how many objects are created.
* Cannot initialize **inside** class/struct definition (if not const)
  + Must initialize **outside** class definition using “<data\_type> Class::<static\_member>” (scope resolution operator)
    - “**outside**” means in the scope where the class is defined (global only, since:)
    - In C++, we **cannot** declare static data member in local classes
    - However, we can define static functions inside class definition (since they don’t require memory allocation)
    - Defining static variable inside the class definition would imply allocating memory for that variable for each object (which is NOT static)
  + Does not need any objects to use that member
  + Can only access the member (from either object or Class) after it is initialized outside.