**Dynamic Memory Allocation**

**What:**

Dynamic memory allocation in C/C++ refers to performing memory allocation manually by programmer. Dynamically allocated memory is allocated on heap and non-static and local variables get memory allocated on stack.

**Why and when:**

Dynamic allocation is required when you don't know the worst case requirements for memory. Then, it is impossible to statically allocate the necessary memory, because you don't know how much you will need. Even if you know the worst case requirements, it may still be desirable to use dynamic memory allocation. It allows for the system memory to be used more efficiently by multiple processes.

**How:**

we can use **library functions** like ***calloc()*** and ***malloc()*** to allocate memory and ***free()*** to deallocater use **operators** like **new** to *allocate* and **delete** to *deallocate*.

**Syntax:**

for **new**,

pointer-variable = new data-type;

(or)

pointer-variable = **new** data-type[size\_of\_memory];

for **delete**,

**delete** pointer-variable;

(or)

**delete** [] pointer-variable;

Difference between new and delete:

1) new is an operator, malloc is a function

2) new calls constructor, malloc doesn't

3) new returns appropriate pointer, malloc returns void \* and pointer needs to typecast to appropriate type.

Note:

1. Double delete can give undefined behaviour.

2. Deleting a pointer which is pointing to NULL has no effect.

**Delete**

The delete is used to give back the allocated memory to the OS. So it can be used for something else.

So its better to have the ptr assigned with nullptr after deletion when we are not moving out of scope immediately.

**Memory Leak:**