- **Dynamic Memory Allocation**

  Dynamic data structure provides flexibility in adding, deleting or rearranging data items at runtime. Dynamic memory management technique permits us to allocate additional memory space or to release unwanted space at runtime.

The process of allocating memory at runtime is known as *Dynamic Memory Allocation.* Although C does not inherit this facility, there are four library routine known as "**Memory Management Function"** these can be used for allocating and freeing memory during program execution.

- **malloc**
- **calloc**
- **free**
- **realloc**

MALLOC

A block of memory may be allocated using the **malloc** function. The **malloc** function is like a request to the RAM of the system to allocate memory, if the request is granted (*i.e., the malloc function stays successful in allocating memory)*, it returns a pointer first block of memory. The type of the pointers it returns is void, which means that we can assign it any type of pointer. However if the **malloc** function fails to allocate the require amount of memory, it returns a NULL.

The general form of **malloc** function is

ptr = (type_cast *) malloc (size);

where **ptr** is a pointer of type *type_cast.* **type_cast** is a data type into which returned pointer (or type void) is to be converted, and **size** specified the size of allocated memory block in bytes.

Example : n = (int *) malloc (5 *sizeof(int));

CALLOC

calloc is another memory allocation function that is normally used for requesting memory space at runtime for storing derive data types such as array and structure. The previous memory allocating function **malloc** allocates a single block of storage space , **calloc** allocates multiple block of storage, each at same size.

The general form of **calloc** function

ptr = (type_cast *) calloc (n,size_type);

the above statement allocates contiguous space for **n** block, each having same size **size_type bytes.**

FREE funtion

The **free** function is used to de-allocate the previously allocated memory using **malloc** and **calloc** functions. When we no longer need the data we store in a block of memory, and we do not intend to use that block for storing any other information, we release that block of memory for future use.

The general form of **free** function is

free(ptr);

REALLOC

This function is used to resize the size of memory block, which is already allocated (*i.e., to modify the size of memory block which is already allocated).* It is used in two situations

- ➢ If the allocated memory block is insufficient for the current application.
- ➢ If the allocated memory is much more than what is required by the current application. In other words, it provides even more precise and efficient utilization of memory.

The general form of **realloc** function

ptr = (type_cast *) realloc (ptr,new_size_type);

where **ptr** is the pointer holding the starting address of the already allocated memory block. But **new_size_type** in bytes you want the system to allocate now, its size may be smaller than previously allocated memory block or greater depending upon the requirement.