Memory Management in C++

Memory Management in C++ -

What is Memory Management?

Memory management is a process of managing computer memory, assigning the memory space to the programs to improve the overall system performance.

Why is Memory Management required?

Arrays store the homogeneous data, so most of the time, memory is allocated to the array at the declaration time. Sometimes the situation arises when the exact memory is not determined until runtime. To avoid the wastage of memory, we can use the new operator to allocate the memory dynamically at the run time.

Memory Management Operators -

In C Language, we are using **malloc(), calloc()** and **realloc()** functions to allocate the memory dynamically at run time along with **free()** function is used to deallocate the allocated memory. C++ supports all these but along with there are two unary operators such as **new** and **delete** to perform the same tasks respectively.

New Operator -

A new operator is used to create the object while a delete operator is used to delete the object. When the object is created by using the new operator, then the object will exist until we explicitly use the delete operator to delete the object.

Syntax - pointer_variable = new data-type

Example –

```
int *p = new int;
float *q =   new float;
```

Assigning a value to the newly created object –

Syntax - pointer_variable = new data-type(value);

Example -

```
int *p = new int(45);
float *p = new float(9.8);
```

How to create a single dimensional array -

Syntax - pointer-variable = new data-type[size];

Example –

```
int *a1 = new int[8];
float *a2 = new float[5];
```

Memory Management in C++ -

Delete Operator -

When memory is no longer required, then it needs to be deallocated so that the memory can be used for another purpose.

Syntax - **delete pointer_variable;**

Example –

```
delete p;
delete q;
```

Delete memory space from array -

Syntax - **delete [size] pointer_variable;**

Example –

```
delete [ ] pointer_variable;
```

Example on Integer variable -

```
#include<iostream.h>
#include<conio.h>

void main()
{
		int *ptr;
		ptr = new int;

		cout<<"Enter the number: ";
		cin>>*ptr;

		cout<<"\nRequired number is: "<<*ptr;
}
```

Example on Array -

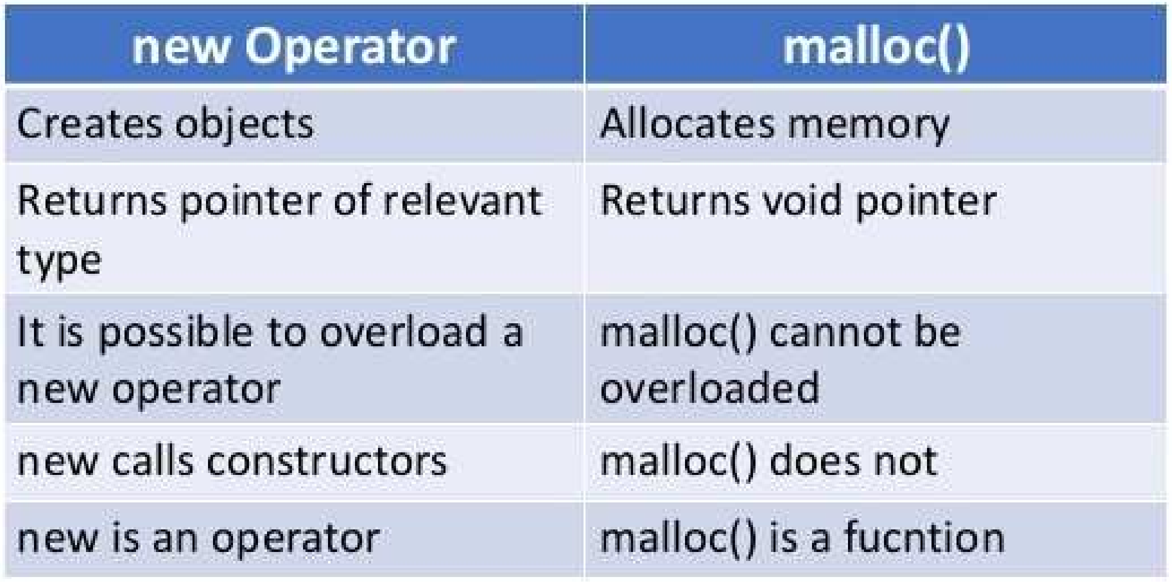
```
#include<iostream.h>
#include<conio.h>

void main()
{
        int i,size;

        cout<<"Enter the size of the array : ";
        cin >> size;

        int *arr = new int[size];

        cout<<"\nEnter the element : ";
        for(i=0;i<size;i++)
        {
                cin>>arr[i];
        }

        cout<<"\nThe elements that you have entered are :";
        for(i=0;i<size;i++)
        {
                cout<<arr[i]<<" ";
        }
        delete arr;
}
```

Advantages of the new operator -

The following are the advantages of the new operator -

1. It does not use the **sizeof()** operator as it automatically computes the size of the data object.
2. It automatically returns the correct data type pointer, so it does not need to use the typecasting.
3. Like other operators, the **new** and **delete** operator can also be overloaded.
4. It also allows you to **initialize the data object** while creating the memory space for the object.

Difference between new and malloc() operator -



new Operator	malloc()
Creates objects	Allocates memory
Returns pointer of relevant type	Returns void pointer
It is possible to overload a new operator	malloc() cannot be overloaded
new calls constructors	malloc() does not
new is an operator	malloc() is a fucntion

Difference between delete and free() operator -

The following are the differences between delete and free() in C++ are:

1. The delete is an operator that de-allocates the memory dynamically while the free() is a function that destroys the memory at the runtime.
2. The delete operator is used to delete the pointer, which is either allocated using new operator or a NULL pointer, whereas the free() function is used to delete the pointer that is either allocated using malloc(), calloc() or realloc() function or NULL pointer.
3. When the delete operator destroys the allocated memory, then it calls the destructor of the class in C++, whereas the free() function does not call the destructor; it only frees the memory from the heap.
4. The delete() operator is faster than the free() function.