Subject Name: **Object Oriented Programming Using C++**

Subject Code: **BCA-301 N**

Subject Topic: **Static and Dynamic Memory Allocation**

Abhishek Dwivedi

Assistant Professor

Department of Computer Application

UIET, CSJM University, Kanpur

Memory Allocation

- **Memory Allocation:** Memory allocation is a process by which computer programs and services are assigned with physical or virtual memory space. The memory allocation is done either before or at the time of program execution. There are two types of memory allocations:
- Compile-time or Static Memory Allocation
- Run-time or Dynamic Memory Allocation

Static Memory Allocation / Dynamic Memory Allocation

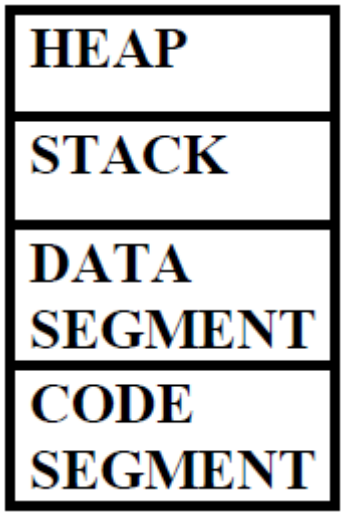
- **Static Memory Allocation:** Static Memory is allocated for declared variables by the compiler. The address can be found using the addressof operator and can be assigned to a pointer. The memory is allocated during compile time.
- **Dynamic Memory Allocation:** Memory allocation done at the time of execution(run time) is known as **dynamic memory allocation**. Functions calloc() and malloc() support allocating dynamic memory. In the Dynamic allocation of memory space is allocated by using these functions when the value is returned by functions and assigned to pointer variables.

Differences between Static Memory Allocation and Dynamic Memory Allocation

Static Memory Allocation	Dynamic Memory Allocation
In static memory allocation, memory is allocated **before the execution** of the program begins.	In Dynamic memory allocation, memory is allocated **during the execution** of the program.
Memory allocation and deallocation actions are not performed during the execution.	Memory allocation and deallocation actions are performed during the execution.
It uses stack for managing the static allocation of memory	It uses heap for managing the dynamic allocation of memory
The data in static memory is allocated permanently.	The data in dynamic memory is allocated only when program unit is active.
It is less efficient	It is more efficient

Dynamic Memory Allocation in C++

A basic memory architecture used by any C++ program:



- **Code Segment**: Compiled program with executive instructions are kept in code segment. It is read only. In order to avoid over writing of stack and heap, code segment is kept below stack and heap.
- **Data Segment**: Global variables and static variables are kept in data segment. It is not read only.

- **Stack**: A stack is usually pre-allocated memory. The stack is a LIFO data structure. Each new variable is pushed onto the stack. Once variable goes out of scope, memory is freed. Once a stack variable is freed, that region of memory becomes available for other variables. The stack grows and shrinks as functions push and pop local variables. It stores local data, return addresses, arguments passed to functions and current status of memory.
- **Heap**: Memory is allocated during program execution. Memory is allocated using new operator and deallocating memory using delete operator.

Allocation of Heap Memory using new Keyword

- How to allocate heap memory to a variable or class object using the **new** keyword.
- Syntax:

  pointervariable = new datatype

Example:

```
int *pv;
pv = new int;
```

Or combine both statements

```
int  *pv = new int;
```

```
int *pv = new int[10];    // allocating block of memory
```

- If enough memory is not available in the **heap** it is indicated by throwing an exception of type std::bad_alloc and a pointer is returned.

Deallocation of memory using delete Keyword

- Once heap memory is allocated to a variable or class object using the new keyword, we can deallocate that memory space using the delete keyword.
- Syntax:

delete pointervariable;

- Example:

delete pv; **//deallocate memory for one element**

delete[] pv; **//deallocate memory for array**

Example

```
#include<iostream.h>
#include<conio.h>
void main()
 {
     int size,i;
      int *ptr;
     cout<<"\n\tEnter size of Array : ";
     cin>>size;
     ptr = new int[size];                    //Creating memory at run-time
     for(i=0;i<size;i++)                     //Input array from user.
     {
        cout<<"\nEnter any number : ";
        cin>>ptr[i];
     }
     for(i=0;i<size;i++)                     //Output array to console.
       {cout<<ptr[i]<<endl;}
     delete[] ptr;                           //deallocating all the memory
getch();
}
```

Dynamic Memory Allocation for Objects

```
class A
{
    public:
        A()
        {
            cout << "Constructor" << endl;
        }
        ~A()
        {
            cout << "Destructor" << endl;
        }
};
 void main()
{
    A  *a = new A[4];
    delete [] a;                        // Delete array
    getch();
 }
```

References:

- www.studytonight.com
- www.tutorialpoint.com
- www.geeksforgeeks.org
- “Object oriented programming in C++”, Robert Lafore
- “Object oriented programming with C++”, E.Balagurusamy