Module-2 : Dynamic Memory Allocation

Static vs Dynamic Memory:

- Stack → Static memory
- Heap → Dynamic memory

Static

- limited memory
- Compile time memory
- Automatic memory clear

Dynamic

- More memory
- Run time memory
- User instructed

Create Dynamic Memory:

When we initialize (**int x = 10**;) it creates a **static memory**. Static memory has a **main functio**n that creates **x variables** (it captures **4 bytes**) and **stores 10** and gives **4 bytes to x** variable. static memory **didn't increase** the size. if we need **more t**hen we **use long long int.**

In dynamic memory, if we need to create new memory we use the **New keyword** (**new data type = new int;).** it can access Heap then we need the data type which one I need

(means int or float so on). if we need **int (data type)** it creates 4 bytes to heap memory. if we don’t delete this it will always exist in the memory. Heap memory gives us the address and the address needs a pointer for the store(**int *a =**). then we will able to access the address. (**int *a = new int;)**

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
   int *a = new int ; //*a means declare
   *a =10; // de references
   cout << a << endl; // aderss print hby
   cout << *a << endl;// value print hby
   return 0;
}
input: 10
output: 10
```

Use Function

```
#include<bits/stdc++.h>
using namespace std;

int * fun()
{
    int *a = new int;
    *a = 100;
    return a;
}
int main()
{
    int *p = fun();
    cout << p << "\n" << *p << endl;
   return 0;
}

output: 0x726800
        100
```

Create Dynamic Array and Return Array From Function:

- Initialize: **new data type [size]; → new int [5];**
- Receive: **int *variable = new data type [size]; → int *a = new int [5];**

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int n;
    cin >> n;

    int * a = new int [n];
    for(int i=0; i<n; i++)
    {
        cin >> a[i];
    }
    for(int i=0; i<n; i++)
    {
        cout << a[i] <<endl;
    }
   return 0;
}
input: 5
       1 2 3 4 5

output: 1
        2
        3
        4
        5
```

```
#include<bits/stdc++.h>
using namespace std;

int *fun()
{
   int *a = new int[5];
   for(int i=0; i<5; i++)
   {
    cin >> a[i];
```

```
   }
   return a;
}
int main()
{
    int *a = fun();
    for(int i=0; i<5; i++)
    {
        cout << a[i] << " ";
    }

   return 0;
}

input: 1 2 3 4 5
output: 1 2 3 4 5
```

Increase the Size of the Array and Delete Dynamic Memory:

if we use to **delete a;** the memory will be deleted .

Increase the Size of the array:

- First, we create memory in a dynamic array with size 3
- Second, we create another memory in a dynamic array with size 3
- then we store data (10 20 30) in both arrays
- then we delete data from the first array
- then we increase the first array size to 5
- then we copy data from the second array to the first array
- then we clear the second array of data
- now we have only one array which is the first array with size 5
- when we copy the second array to the first array we have 2 more sizes are available now we can add two more data in our array

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
    int *a = new int [3];
    int *b = new int [3];

    for(int i=0; i<3; i++)
    {
        cin >> a[i];
        b[i] = a[i];
    }
    delete[] a;

    a = new int [5];
    for(int i=0; i<3;i++)
    {
        a[i] = b[i];
    }
    delete[] b;

    a[3] = 4;
    a[4] = 5;
    for(int i=0; i<5;i++)
    {
        cout << a[i] << " ";
    }

    return 0;
}

input: 1 2 3
output: 1 2 3 4 5
```