Dynamic Memory Allocation

Using array in programming, we allocate a fixed size for our data. This size can’t be increased or decreased while execution of the program. We can’t change it even if the size allocated is more or less than our requirement. This type of allocation of memory is called **Static Memory Allocation**. This leads to wastage or shortage of memory.

Fortunately, C allows programmer to allocate memory dynamically i.e. during run time and this process is called dynamic memory allocation. By allocating memory dynamically, we can use only the amount of memory required for us.

For this, C has four built in functions under **“stdlib.h”** header files for allocating memory dynamically.

Dynamic memory allocation in c language is possible by 4 functions of stdlib.h header file.

1. malloc()
2. calloc()
3. realloc()
4. free()

The difference between static memory allocation and dynamic memory allocation.

static memory allocation	**dynamic memory allocation**
memory is allocated at compile time.	memory is allocated at run time.
memory can't be increased while executing program.	memory can be increased while executing program.
used in array.	used in linked list.

malloc() function in C

- The malloc() function allocates single block of requested memory.
- It doesn't initialize memory at execution time, so it has garbage value initially.
- It returns NULL if memory is not sufficient.
- The syntax of malloc() function is given below:

Syntax of malloc

<pointer_variable> = (data_type*)malloc(size);

Example:

Ptr=(int*)malloc(n*sizeof(int));

Example #1 : C program to accept n number and display it using malloc function.

```
#include <stdio.h>

#include <conio.h>

#include <stdlib.h>

void main()

{

        int i,n;

        int *p;

   printf("Enter the value");

   scanf("%d",&n);

   p=(int*)malloc(n*sizeof(int));

   for(i=0;i<n;i++)

   {

        scanf("%d",p+i);


   }

   for(i=0;i<n;i++)

   {

        printf("%d",p[i]);

   }

   getch();

}
```

calloc() function in C

- The calloc() function allocates multiple block of requested memory.
- But the memory allocated by calloc is divided into small equal sizes while memory allocated by malloc is not divided.
- It initially initialize all bytes to zero.
- It returns NULL if memory is not sufficient.
- The syntax of calloc() function is given below:

<pointer_variable> = (data_type*)calloc(n,size);

Example:

Ptr = (float*)calloc(n,sizeof(float);

Example #1 : C program to accept n number and display it using calloc function.

```
#include <stdio.h>

#include <conio.h>

#include <stdlib.h>

void main()

{

        int i,n;

  int *p;

  printf("Enter the value");

  scanf("%d",&n);

  p=(int*)calloc(n,sizeof(int));

  for(i=0;i<n;i++)

  {

        scanf("%d",p+i);


  }

  for(i=0;i<n;i++)

  {
```

```
            printf("%d\t",p[i]);
  }
  getch();
}
```

realloc() function in C

If memory is not sufficient for malloc() or calloc(), you can reallocate the memory by realloc() function. In short, it changes the memory size.

Let's see the syntax of realloc() function.

```
<pointer_variable> =(data_type*) realloc(old_pointer_variable,newsize);
```

Example

```
Ptr=(data_type*)realloc(ptr,100);
```

Example #1 : C program to accept n number and display it using malloc function and realloc function.

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
void main()
{

            int *ptr;
            int n, i;

            printf("Enter number of elements:\n");
   scanf("%d",&n);
                        // Dynamically allocate memory using calloc()
                        ptr = (int*)malloc(n*sizeof(int));
                        // Get the elements of the array
                        for (i = 0; i < n; i++)
```

```
{
                scanf("%d",ptr+i);
        }
        // Print the elements of the array
        printf("The elements of the array are: ");
        for (i = 0; i < n; i++) {
                printf("%d\t", ptr[i]);
        }

        // Get the new size for the array
        printf("\n\nEnter the new size of the array: \n");
scanf("%d",&n);
        // Dynamically re-allocate memory using realloc()
        ptr = (int*)realloc(ptr, n * sizeof(int));

        // Memory has been successfully allocated
        printf("Memory successfully re-allocated using realloc.\n");

        // Get the new elements of the array
        for (i = 0; i < n; i++) {
                scanf("%d",ptr+i);
        }

        // Print the elements of the array
        printf("The elements of the array are: ");
        for (i = 0; i < n; i++)
{
                printf("%d\t ",ptr[i]);
        }
```

```
                free(ptr);

        getch();

}
```

free() function in C

The memory occupied by malloc() or calloc() functions must be released by calling free() function. Otherwise, it will consume memory until program exit.

Let's see the syntax of free() function.

free(<pointer_name>);

example:

free(ptr);