Dynamic Memory Allocation

C – Pointer

Pointers in C language is a variable that stores/points the address of another variable. A Pointer in C is used to allocate memory dynamically i.e. at run time. The pointer variable might be belonging to any of the data type such as int, float, char, double, short etc.

Pointer Syntax : data_type *var_name; Example : int *p; char *p;

Where, * is used to denote that "p" is pointer variable and not a normal variable.

KEY POINTS TO REMEMBER ABOUT POINTERS IN C:

Normal variable stores the value whereas pointer variable stores the address of the variable.

The content of the C pointer always be a whole number i.e. address.

Always C pointer is initialized to null, i.e. int *p = null.

The value of null pointer is 0.

& symbol is used to get the address of the variable.

* symbol is used to get the value of the variable that the pointer is pointing to.

If a pointer in C is assigned to NULL, it means it is pointing to nothing.

Two pointers can be subtracted to know how many elements are available between these two pointers.

But, Pointer addition, multiplication, division are not allowed.

The size of any pointer is 2 byte (for 16 bit compiler).

EXAMPLE PROGRAM FOR POINTERS IN C:

```
#include <stdio.h>
int main()
{
   int *ptr, q;
   q = 50;
   /* address of q is assigned to ptr */
   ptr = &q;
   /* display q's value using ptr variable */
   printf("%d", *ptr);
   return 0;
}
```

POINTERS IN C – PROGRAM OUTPUT:

```
50
```

DYNAMIC MEMORY ALLOCATION IN C:

The process of allocating memory during program execution is called dynamic memory allocation.

DYNAMIC MEMORY ALLOCATION FUNCTIONS IN C:

C language offers 4 dynamic memory allocation functions. They are,

malloc()

calloc()

realloc()

free()

Function	Syntax
malloc ()	malloc (number *sizeof(int));
calloc ()	calloc (number, sizeof(int));
realloc ()	realloc (pointer_name, number * sizeof(int));
free ()	free (pointer_name);

1. MALLOC() FUNCTION IN C:

malloc () function is used to allocate space in memory during the execution of the program.
malloc () does not initialize the memory allocated during execution. It carries garbage value.
malloc () function returns null pointer if it couldn't able to allocate requested amount of memory.

EXAMPLE PROGRAM FOR MALLOC() FUNCTION IN C:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main()
{
    char *mem_allocation;
    /* memory is allocated dynamically */
    mem_allocation = (char *) malloc( 20 * sizeof(char) );
    if( mem_allocation== NULL )
    {
      printf("Couldn't able to allocate requested memory\n");
    }
    else
    {
      strcpy( mem_allocation,"Programming and Data Structures");
    }
    printf("Dynamically allocated memory content : " \ "%s\n", mem_allocation );
    free(mem_allocation);
}
```

OUTPUT:

```
Dynamically allocated memory content : Programming and Data Structures
```

2. CALLOC() FUNCTION IN C:

calloc () function is also like malloc () function. But calloc () initializes the allocated memory to zero. But, malloc() doesn't.

EXAMPLE PROGRAM FOR CALLOC() FUNCTION IN C:

```
#include <stdio.h>
```

```
#include <string.h>
#include <stdlib.h>

int main()
{
    char *mem_allocation;
    /* memory is allocated dynamically */
    mem_allocation = (char *) calloc( 20, sizeof(char) );
    if( mem_allocation== NULL )
    {
      printf("Couldn't able to allocate requested memory\n");
    }
    else
    {
       strcpy( mem_allocation,"Programming in C");
    }
       printf("Dynamically allocated memory content   : " \
             "%s\n", mem_allocation );
       free(mem_allocation);
}
```

OUTPUT:

```
Dynamically allocated memory content : Programming in C
```

3. REALLOC() FUNCTION IN C:

realloc () function modifies the allocated memory size by malloc () and calloc () functions to new size.

If enough space doesn't exist in memory of current block to extend, new block is allocated for the full size of reallocation, then copies the existing data to new block and then frees the old block.

4. FREE() FUNCTION IN C:

free () function frees the allocated memory by malloc (), calloc (), realloc () functions and returns the memory to the system.

EXAMPLE PROGRAM FOR REALLOC() AND FREE() FUNCTIONS IN C:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main()
{
   char *mem_allocation;
   /* memory is allocated dynamically */
   mem_allocation = (char *) malloc( 20 * sizeof(char) );
   if( mem_allocation == NULL )
   {
      printf("Couldn't able to allocate requested memory\n");
```

```
    }
    else
    {
      strcpy( mem_allocation,"Programming in C");
    }
    printf("Dynamically allocated memory content  : " \
          "%s\n", mem_allocation );
    mem_allocation=realloc(mem_allocation,100*sizeof(char));
    if( mem_allocation == NULL )
    {
       printf("Couldn't able to allocate requested memory\n");
    }
    else
    {
       strcpy( mem_allocation,"space is extended upto " \ "100 characters");
    }
    printf("Resized memory : %s\n", mem_allocation );
    free(mem_allocation);
 }
```

OUTPUT:

```
Dynamically allocated memory content : Programming in C
Resized memory : space is extended upto 100 characters
```

DIFFERENCE BETWEEN STATIC MEMORY ALLOCATION AND DYNAMIC MEMORY ALLOCATION IN C:

Static memory allocation	**Dynamic memory allocation**
In static memory allocation, memory is allocated while writing the C program. Actually, user requested memory will be allocated at compile time.	In dynamic memory allocation, memory is allocated while executing the program. That means at run time.
Memory size can’t be modified while execution. Example: array	Memory size can be modified while execution. Example: Linked list

DIFFERENCE BETWEEN MALLOC() AND CALLOC() FUNCTIONS IN C:

malloc()	**calloc()**
It allocates only single block of requested memory	It allocates multiple blocks of requested memory
int *ptr; ptr = (int*) malloc(20 * sizeof(int));For the above, 20*4 bytes of memory only allocated in one block. Total = 80 bytes	int *ptr; ptr = calloc(20, 20 * sizeof(int));For the above, 20 blocks of memory will be created and each contains 20*4 bytes of memory.

	Total = 1600 bytes
malloc () doesn’t initializes the allocated memory. It contains garbage values	calloc () initializes the allocated memory to zero
type cast must be done since this function returns void pointer int *ptr;ptr = (int*)malloc(sizeof(int)*20);	Same as malloc () function int *ptr;ptr = (int*)calloc(20, 20 * sizeof(int));