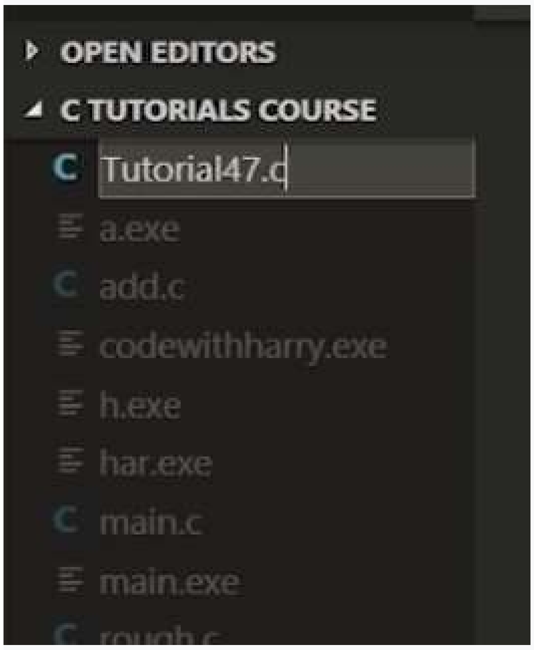
Dynamic Memory Allocation Malloc Calloc Realloc & Free(): C Tutorial In Hindi #47

The selection of right IDE for programming is one of the most important thing. To run a **C program**, we need an IDE's like Visual Studio Code or Codeblocks. For this series, we are using Virtual Studio Code (VS Code). It provides the tools that a developer needs for a quick **code**-build-debug cycle. To download VS Code, click on ***Download Virtual Studio Code*** . For guidance, check the tutorial ***Install & Configure VS Code.*** Now open VS Code and create a file with name **"tutorial47.c".**



In the previous tutorial, we learned about memory allocation and its segments and a little about dynamic memory. This tutorial will cover dynamic memory allocation in detail and its practical implementation on Visual Studio. As discussed in the previous tutorial i.e., ***Tutorial #45***, dynamic memory is allocated at the run time. So in this tutorial, I will show you how we can allocate memory dynamically using a heap data structure.

For the allocation of memory using the heap, we have four functions:

- Malloc
- Calloc
- Realloc
- Free

Code as described/written in the video

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

int main()
{
    // //Use of malloc
    // int *ptr;
    // int n;
    // printf("Enter the size of the array you want to create\n");
    // scanf("%d", &n);

    // ptr = (int *)malloc(n * sizeof(int));
    // for (int i = 0; i < n; i++)
    // {
    //     printf("Enter the value no %d of this array\n",i);
    //    scanf("%d", &ptr[i]);
    // }

    // for (int i = 0; i < n; i++)
    // {
    //     printf("The value at %d of this array is %d\n",i, ptr[i]);
    // }

    //Use of calloc
    int *ptr;
```

malloc():

malloc stands for memory allocation. As can be guessed by its name, it requests memory from the heap and returns a pointer to the memory. The pointer is of the void type, so that we can typecast it for any variables. All the values at the allocation time are initialized to garbage values. Its syntax is simple as we have to provide the memory space along with the size we want in bytes.

Syntax:

```
ptr = (ptr-type*) malloc(size_in_bytes)
```

For example:

```
int *ptr;
ptr = (int*) malloc (3* sizeof(int))
```

Note: We are using the sizeof() function here because the size of int may differ in different systems, so to be on the safe side.

calloc():

calloc stands for contiguous allocation. It also requests memory from the heap and returns a pointer to the memory and has the same functionality as malloc(), two main differences, though. We have to send as parameters the number of blocks needed along with their size. The second difference is a major one. That is, in calloc(), the values at the allocation time are initialized to 0 instead of garbage value.

Syntax:

```
ptr = (ptr-type*) calloc(n,size_in_bytes)
```

For example:

```
int *ptr;
ptr = (int*) malloc (10, sizeof(int))
```

realloc():

realloc stands for reallocation. It is used in cases where the dynamic memory is insufficient or wants to increase the already allocated memory to store more data. Its syntax is simple as we just have to overwrite the memory already allocated as a parameter in the function while providing the data related to the pointer.

Syntax:

```
ptr = (ptr-type*) realloc(ptr,new_size_in_bytes)
```

For example:

```
ptr = (int*) realloc (ptr, 5* sizeof(int))
```

free():

As we discussed earlier, while discussing the disadvantages of dynamic memory allocation that we have to free up the allocated memory space manually as there is no automatic procedure for that. So free is used to free up the space occupied by the allocated memory. Its syntax is the easiest of all, as we have to send the pointer as a parameter inside the function.

Syntax:

```
free(ptr)
```

Summary:

We use dynamic memory to make the program more efficient as we can define the memory allocation ourselves, thus reducing the program's size, which will have a direct effect on its execution timing. We can allocate dynamic memory fully by using only the four functions given above. Dynamic memory allocation has its advantages and disadvantages, but in order to become a great programmer, we must learn all the concepts completely.