Date
Page

* Static and Dynamic memory allocation -

i) Static memory allocation -
The process of allocating memory during compile time is known as static memory allocation.

ii) Dynamic memory allocation -
The process of allocating memory during run time ~~to~~ or execution time as required is known as dynamic memory allocation.

* Difference b/w static and dynamic memory allocation -

Static	Dynamic
i) Memory is allocated during compile time.	i) Memory is allocated during run time.
ii) Amount of memor allocated is fixed, can't change.	ii) Amount of memory allocated can be changed
iii) Wastage of memory may occure.	iii) No wastage of memory
iv) Memory is allocated from stack segment.	iv) Memory is allocated from heap segment.

Date
Page

* Dynamic memory allocation function -

i) malloc()
ii) calloc()
iii) realloc()
iv) free()

The following functions are used to allocate and deallocate the memory from segment dynamically.

* These functions are defined in header file "alloc.h" or "stdlib.h".

a) malloc() function -

It is used to allocate a single block of memory of specified size dynamically and returns a pointer to the allocated block. The initial values in all the memory locations will be garbage value.

Syntax - datatype * ptr = (data type *) malloc(required amount of memory);

Eg:- int * ptr;
ptr = (int *) malloc(10 * sizeof(int));

(It allocates 10 * sizeof(int) = 10 * 4 bytes = 40 b; and starting address of block is assigned to pointer ptr.)

```
# include <stdio.h>
# include <stdlib.h>
int main()
 {
    int n, i, *ptr;
  printf("Enter total no. of values");
    scanf("%d", &n);
  ptr = (int *)malloc(n * sizeof(int));

  printf("\n Enter the values:");
     for(i=0; i<n; i++)
          scanf("%d", (ptr+i));
  printf("\n The entered values are:");
     for(i=0; i<n; i++)
          printf("%d", *(ptr+i));

    free(ptr);
}
```

ii) Calloc() function (contiguous allocations) -

It is used to allocate multiple blocks memory of specified size dynamically and returns a pointer to the allocated block. The initial values in all the memory allocation will be zero.

Syntax -

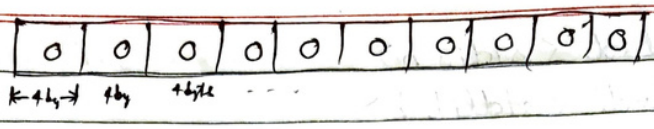
datatype *Ptr = (data type *)calloc(No.of blocks, required amount of memory);

Eg:- int *ptr;
ptr = (int *)calloc(10, sizeof(int));

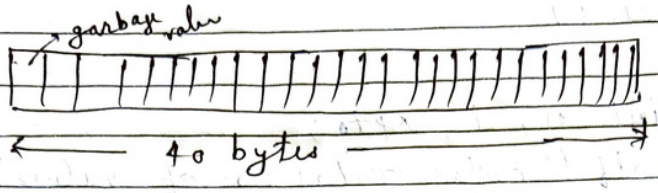
(It allocates 10 block of memory. Each block will be of size of 4 bytes, so total 40 bytes of memory is allocated)

Date
Page

* Using calloc()



* Using malloc()



*
```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int n, i, *ptr;
    printf("Enter total no. of values: ");
    scanf("%d", &n);

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

    printf("\n Enter the values: ");
    for(i = 0; i<n; i++)
        scanf("%d", (ptr+i));

    printf("\n The entered values are: ");
    for(i = 0; i<n; i++)
        printf("%d \t", *(ptr+i));

    free(ptr);
}
```

iii) realloc() function -

It is used to increase or decrease the size of an already allocated memory using malloc or calloc function and returns a pointer to the newly allocated block.

syntax -

datatype *ptr = (data type *)realloc(ptrname, New amount of memory);

Ex:- int * ptr;
ptr = (int *) malloc(10 * sizeof(int));
(it allocates 40 bytes of memory)

ptr = (int *) realloc(ptr, 20);
(it decreases the 40 bytes of memory to 20 bytes)

iv) free() function -

It is used to deallocate or release the memory which was already allocated by using malloc, calloc or realloc function.

syntax -

free(pointer name);

Ex:- char * ptr;
ptr = (char *) malloc(5 * sizeof(char));
free(ptr);