V.V.P Engineering College
Computer Engineering Department
Subject: Programming and Problem Solving

Chapter no: 9 Dynamic Memory Allocations

What is Dynamic Memory Allocation?
Answer:

The concept of dynamic memory allocation in c language enables the C programmer to allocate memory at runtime. Dynamic memory allocation in c language is possible by 4 functions of stdlib.h header file.

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

Before learning above functions, let's understand 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.

Now let's have a quick look at the methods used for dynamic memory allocation.

malloc()	allocates single block of requested memory.
calloc()	allocates multiple block of requested memory.
realloc()	reallocates the memory occupied by malloc() or calloc() functions.

free()	frees the dynamically allocated memory.

Explain malloc() with example.

Answer:

We use the malloc function to allocate a block of memory of specified size.
This function returns a pointer of type void so, we can assign it to any type of pointer variables.
The malloc function will return NULL if it fails to allocate the required memory space.

Malloc syntax:

Following is the syntax of the malloc function.

```
ptr = (cast_type *) malloc (byte_size);
```

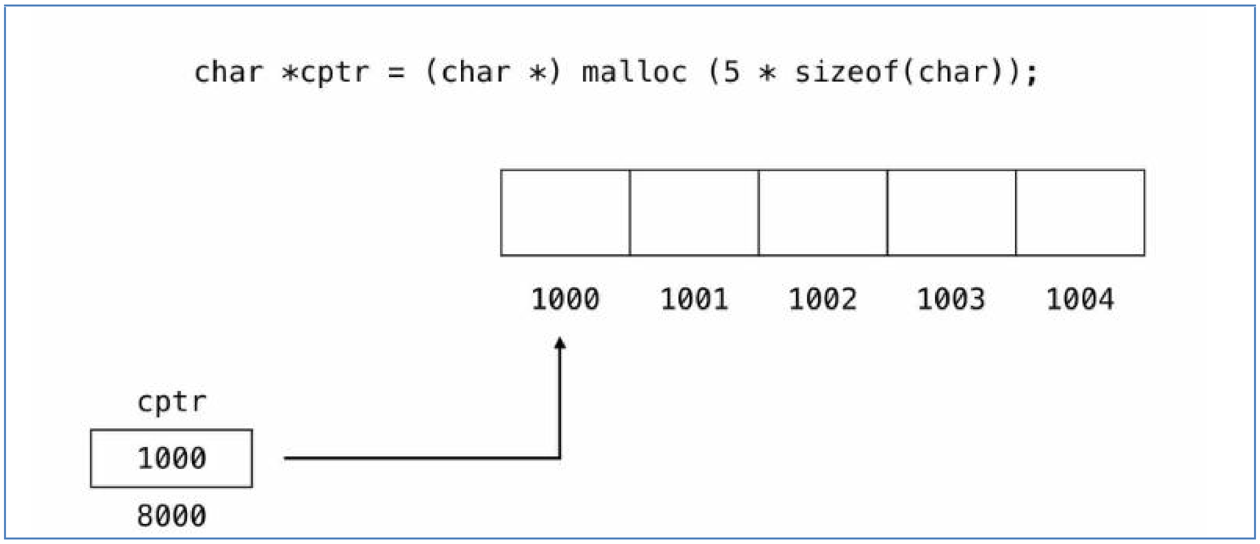
Where, ptr is a pointer variable of type cast_type and byte_size is the memory size that we want to allocate.

Malloc example:

In the following example we are allocating memory space of size 5 bytes to store 5 characters.

```
// char pointer

char *cptr;



// allocate memory

cptr = (char *) malloc (5 * sizeof(char));
```

We can represent this in memory as follows.



We are first computing the byte_size i.e., (5 * sizeof(char)).

Note! sizeof(char) gives us 1 byte and we want to save 5 characters so, total memory space needed is 5x1 = 5 bytes. So, byte_size for this example is 5.

We are passing 5 to the malloc function and on successfully allocating the required memory space it returns a void pointer which we cast into char type pointer by writing (char *).

Then we are assigning the first address of the allocated memory space to a character pointer variable cptr.

3. Explain calloc() with example.
Answer:

We use the calloc function to allocate memory at run time for derived data types like arrays and structures.

Using calloc function we can allocate multiple blocks of memory each of the same size and all the bytes will be set to 0.
This is different from the malloc function from the previous tutorial which is used to allocate single block of memory space.

Calloc syntax:

Following is the syntax of the calloc function to allocate memory dynamically.

```
ptr = (cast_type *) calloc (n, element_size);
```

Where, ptr is a pointer of type cast_type.
n is the total block of contiguous spaces, each of size element_size to be allocated using the calloc function.

Calloc example:

In the following example we are allocating memory space for 3 student structure variable.

```
// student structure
struct student {
  char id[10];
  char firstname[64];
  char lastname[64];
  int score;
};

// new type
typedef struct student candidate;

// student structure pointer
candidate *sptr;

// variables
int no_of_students = 3;

// allocate memory blocks
sptr = (candidate *) calloc (no_of_students, sizeof(candidate));
```

In the above code we are allocating 3 blocks of memory space each of size candidate structure i.e., 140 bytes.