C_132 ⇒ Introduction to Dynamic Memory Allocation in C

SMA / DMA

* Dynamic Memory Allocation → Memory allocated at run time.

* Static Memory Allocation → Memory not allocated at compile time.

* But at compile time; not like memory allocated at compile time.

* At compile time our source code is converted into object code, so we cannot say that memory allocated at compile time.

```
int main()
  {
     int a,b;
     printf("Enter value of a&b:");
     scanf("%d %d", &a,&b);
     printf("a = %d  b=%d", a,b);
     return 0;
  }
```

* Because once the program goes into main memory at that time the compilation starts which converts our source code to object code; then linker & loader will convert our object code (.obj file) to (.exe) code (i.e) executable code which is then put into the RAM and after this only memory is allocated for the variables eg: 'a' and 'b' (i.e) fixed

*So at compile time; the memory allocation for 'a' & 'b' is fixed as 4 bytes for each since it is integer. and this is called static Memory Allocation and hence we cannot change the memory allocated for 'a' & 'b' at run time

*So the memory when allocated at run time which is not going to fixed and that allocated memory can be modified according to our programming needs and it is called Dynamic Memory Allocation

↳ We can increase or decrease memory space based on our programming needs. in DMA

↳ We cannot increase or decrease memory space based on our programming needs in SMA.

Eg:- int a[5]; → fixed memory.

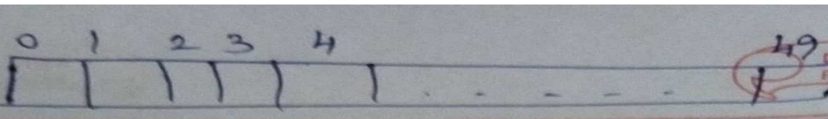
0	1	2	3	4
1	3	-1		

This is at compile time → 5 bytes = 20 bytes is allocated

```
a[
Scanf("%d", &a[0]);
Scanf("%d", &a[1]);
Scanf("%d", &a[2]);
```

Now we are getting input from user at run time

*So we get input from user at run time for only 3 values and remaining 2 spaces are fixed. Suppose you get more than 5 values then it is not allowed since it is SMA.



* Eg:: char str[50]; → memory is allocated 50 bytes and it is fixed

↳ Now at run time if we give value as Sonu or Jenny or Pradeep which has allocated only few bytes of fixed memory and the remaining memory is wasted.

↳ Now how the remaining memory at run time cannot be freed (ie) is SMA memory cannot be re-used; the memory is left free only after exit of the program

* So we should take care of about wastage of memory because when we write a very large programs then this type of memory wastage will lead to give null values or the program will give undefined behaviour

* So we should have a good way of writting a program with the concept of this SMA and DMA to avoid wastage of memory which will be useful when we are computing a large task or creating high end software application.

NOTE :- Drawbacks of SMA

* We cannot increase or decrease the memory in SMA since the memory is fixed * SMA cannot handle memory wastage.

* To remove the drawbacks of SMA we have the concept of DMA.

Advantage of DMA:-

* Memory will be allocated at run time and the memory can be updated/modified based on the programming needs, hence there wont be any wastag of memory.

Functions for DMA are:-

↳ malloc

↳ Calloc

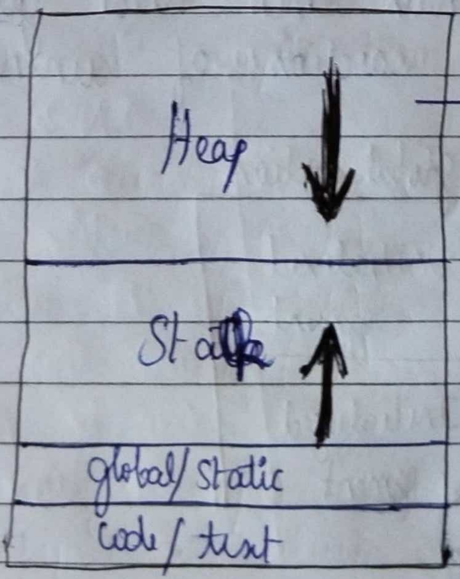
↳ realloc

↳ free

* Before discussing about this functions we will see about how the memory is allocated for our program.

<u>Memory sections</u>

Stack and heap will always grow in opposite direction



→ It is like free pool of memory. (Dynamic memory allocation is done from Heap)

↓

We can allocate or deallocate this memory

Stack and Heap:-

* Heap is like a free pool of memory and the Dynamic Memory allocation is done only from here. So we can allocate or deallocate this type of memory.

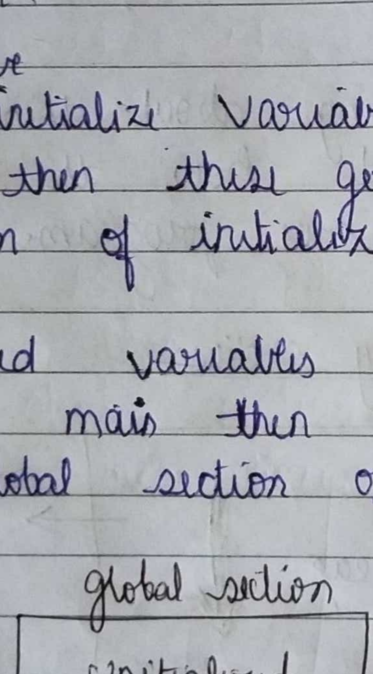
* From the whole memory, stack section will take only limited space, all the local variables and functions will take this stack section memory.

* Always the stack and heap section will grow in opposite section.

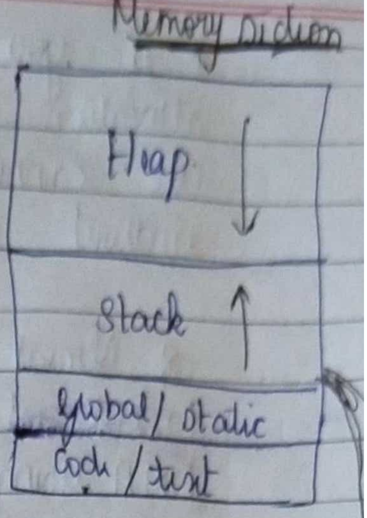
global section:

* If we have initialize variables & functions defined outside the main then these get stored in the global section of initialized segment

* Uninitialized variables & function defined outside the main then these get stored in the global section of uninitialized segment.

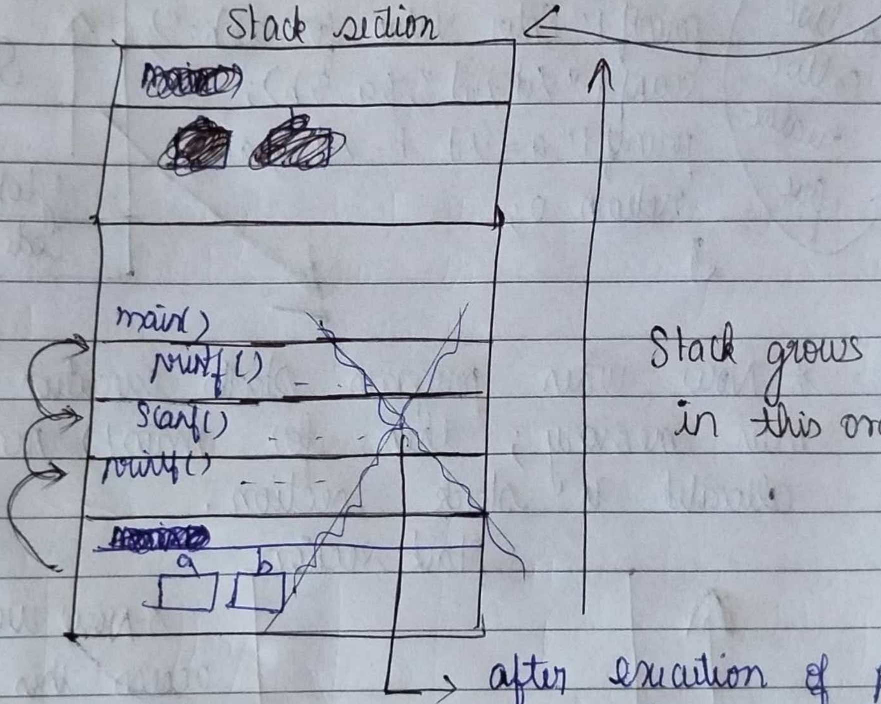


Example for Static Memory allocation of Memory section of our Program



```
Eg:: int i=10;
     int a;
     int main()
     {
        int a,b;
        printf("Enter value a&b");
        scanf("%d %d", &a, &b);
        printf("a=%d b=%d", a,b);
        return 0;
     }
```

* Now when program starts execution from main memory; then for main() the memory is allocated in stack section.



after execution of pgm, once it exit from main function; all the space is now freed up

* This is the way of Static Memory allocation (ie) at Compile time our obj file gets converted to .exe file and get loaded into main memory for execution and hence the memory here is fixed and we cannot change.

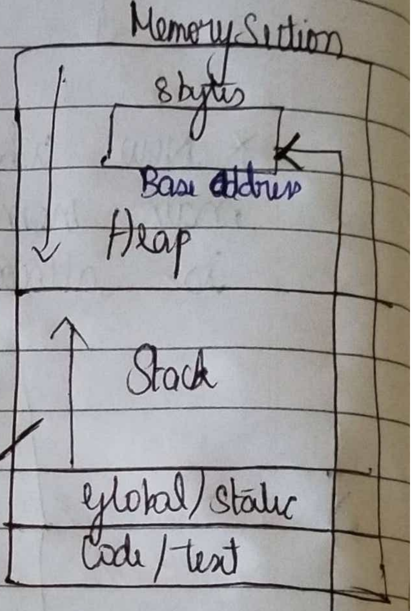
Example for Dynamic Memory Allocation of Memory Section of our Program.

* If you want to use ~~static~~ Dynamic memory allocation then we use pointers. So We cannot use Dynamic Memory Allocation without the help of pointers.

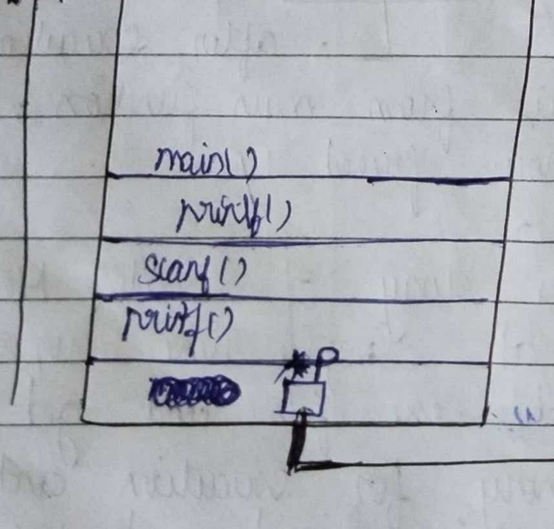
* Almost every Data Structures like Stack, Linked List, Queue, Tree uses the concept of DMA using pointers.

using malloc calloc realloc free

```
eg:  int i=10;
       int a;
     int main()
     {
       int *p;
     printf("Enter value for a&b:");
     scanf("%d%d",&a,&b);
     printf("a=%d b=%d",a,b);
     return 0;
     }
```



* Now when program starts execution from main memory; then for main() memory is allocated in stack section.



* Now we can access this heap section with the DMA using pointers and this section can be modified any time

NOTE

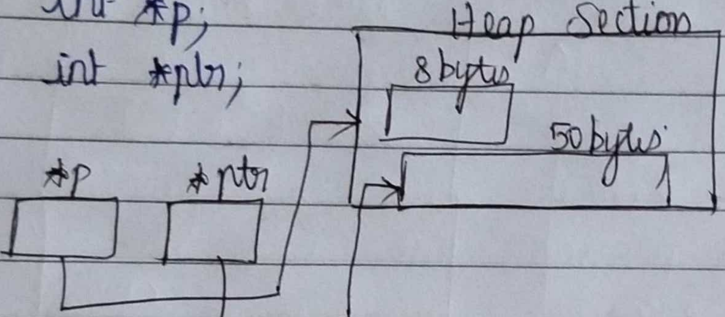
* You have to free the memory once we have done our program execution over. This is very very important.

free() memory

* If we are not freeing this heap memory then this leads to exhausted of memory or memory goes increasing leading to some undefined behaviour of our program.

Eg:: when we have another pointer *ptr to access heap memory of 50 bytes then now the memory here is exhausted.

```
int *p;
int *ptr;
```



* So we can free the pointer P and then we can use another pointer to use the block and then again free the memory and this is the concept of memory reusability of DMA.

* Use of malloc, calloc, realloc and free functions are used in DMA which will be learning in next lessons.