Dynamic Memory Allocation

Static Memory

- So far whatever memory was used is static memory which will be either in stack, data or code segment.
- In static memory the size is fixed, cannot modify, exchange or delete the memory.
- Static memories are called as named location

Dynamic Memory

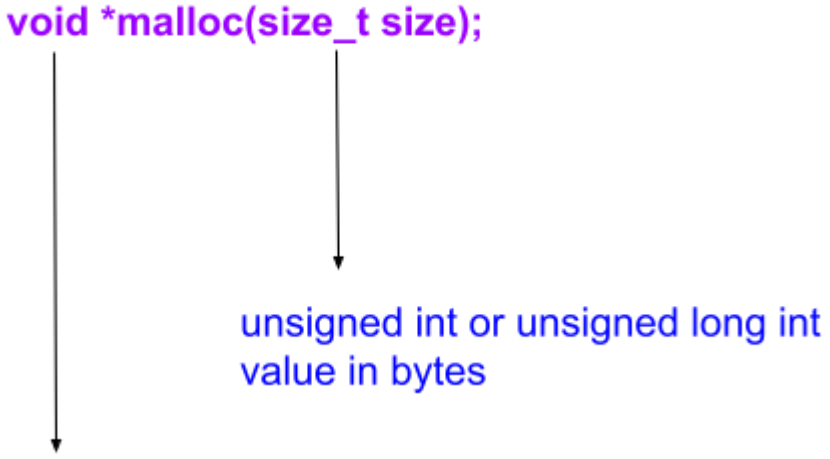
- Memory will be allocated in heap.
- Dynamic memory can be modified, extended or deleted whenever it is required.
- Dynamic memory is managed with the help of pointers and functions like malloc, calloc, realloc and free.
- Because dynamic memory is an unnamed location, to control them precisely we need pointers.
- All the functions are part of stdlib.h

Example:

```
int main()

{

        //dynamic memory allocation 100 byte

        // some operation

        //free the memory
}
```

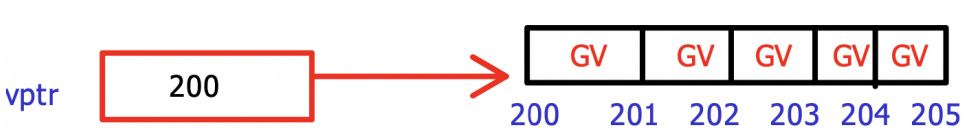
1. Malloc()

- Prototype of the function - void * malloc(size_t size)



void *vptr;

vptr = malloc(5);



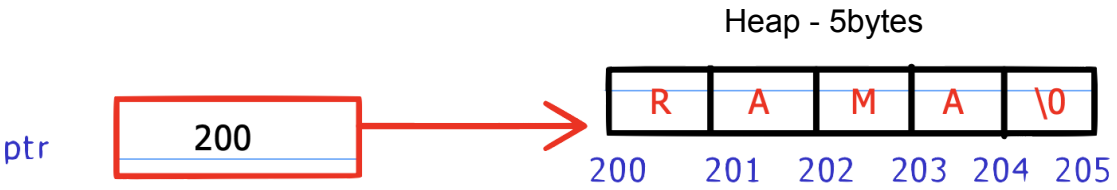
- Malloc will always search for 5 bytes of contiguous memory in heap segment
- If continuous memory bytes are not available then malloc will return NULL
- By default it will be having garbage value

Example usage:

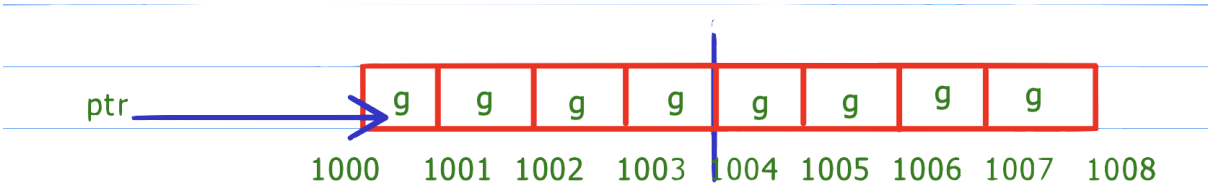
int *ptr;

ptr = malloc(5); //implicit type casting from void * to int *

strcpy(ptr, “RAMA”);



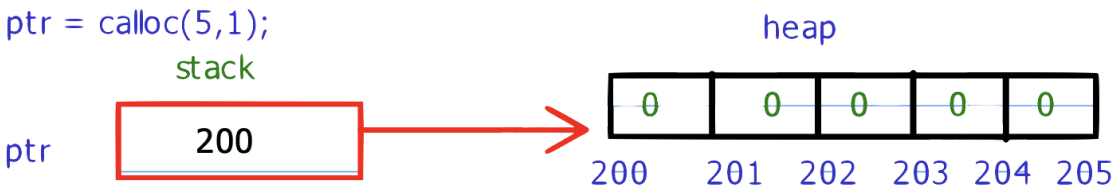
ptr = malloc(2 * sizeof(int));



2. Calloc()

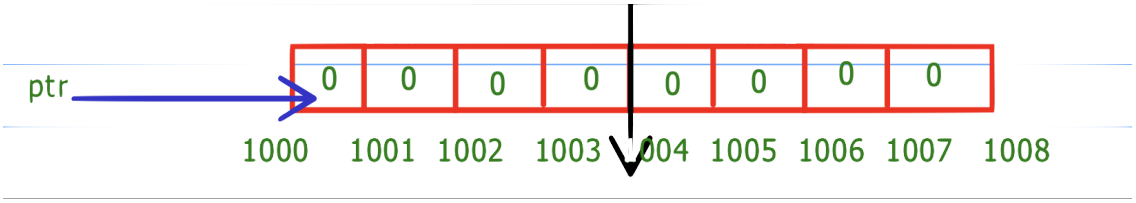
- Prototype of calloc - void *calloc(size_t nmemb, size_t size);
  Where, nmemb - number of member or number of elements or number of blocks
  Size - size of each member

  Example,
  int *ptr;



- Calloc will always search for 5 bytes of contiguous memory in heap segment
- If continuous memory bytes are not available then malloc will return NULL
- By default it will be initialised with 0

Ptr = calloc(2,sizeof(int));



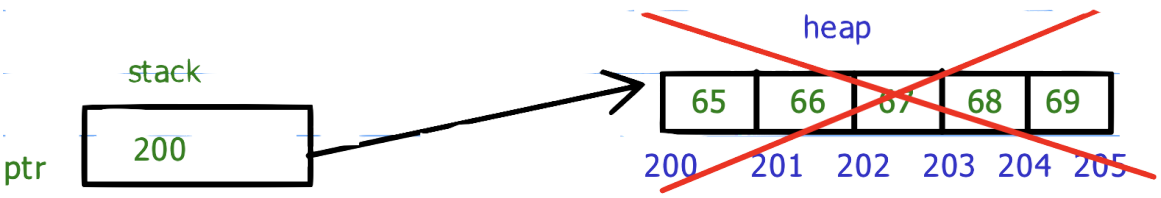
Malloc vs calloc

- With respect to space the same works in a similar way, where both the function allocates contiguous memory in heap.
- When it comes to time calloc takes more time than malloc because of initialization of each block to 0
- Malloc is better with structure and basic memory allocation
- Calloc is better with array because of initialisation to 0

3. Free()

- Free function is used whenever user wants to free/ deallocate unused memory
- free is used to deallocate the dynamically allocated memory by malloc or calloc function
- Prototype -

  void free(void *ptr);

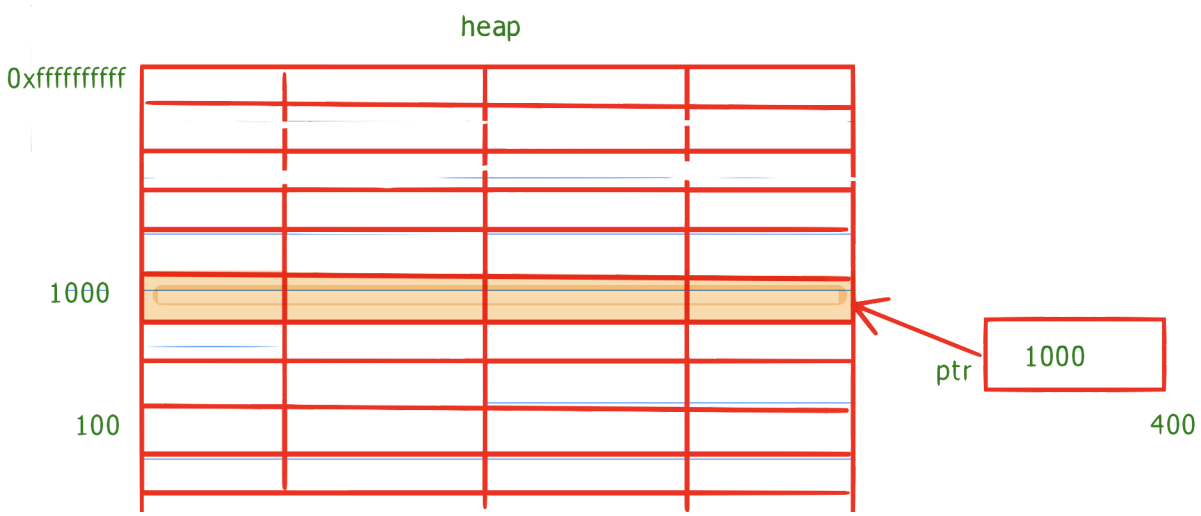


- although heap is deleted from free function, the address which is freed will be available in the pointer
- Pointer which still holds onto the already freed memory is known as a dangling pointer. It is recommended to make dangling pointer to point to Null to avoid undefined behaviour at later

4. Realloc function

- Realloc is used to extend or shrink the previously allocated memory whenever required
- Prototype :
    - void *realloc(void *ptr, size_t size);

Consider void *ptr = malloc(4);



1. realloc(ptr,3) //shrinking the memory
    a. Here, 1 byte will be freed from the pointer.
    b. Legally 3 bytes can be accessed.

2. realloc(ptr,10);
    a. When the size is more than the previous allocation then the realloc will extend the remaining bytes
    b. Here, 7 bytes will be extended.
    c. If 7 bytes are available continuously in the same location then realloc extends the memory and returns the same address.
    d. Else if 7 bytes is not available in the same location then realloc search for whole 10 bytes in some other location and returns the newly allocated memory.
    e. If realloc() fails to expand memory as requested then **it returns NULL , the data in the old memory remains unaffected**.