**Static Memory Allocation**

* **The memory allocated during compilation time is called static memory allocation.**
* **The memory allocated during compile time is fixed and cannot be increased or decreased during run time.**
* **Let us see an example of an array**

**void main()**

**{**

**int arr[5] ={1,2,3,4,5};**

**}**

**Here the we have declared an array of size [5] and we cannot increase the size or decrease the size at runtime. The memory is allocated at compile time and is fixed.**

**Problem during static memory allocation: -**

* **If you are allocating memory of an array/variable during compile time then you have to fix the size at the time of declaration. Size is fixed and user cannot change increase or decrease the size of variable/array during runtime.**
* **If the values stored by the user in the array/variable at run time is less then the size specified then there will be the wastage of memory.**
* **If the value stored by the user in the array/ variable at run time is more then the size of specified then the program may crash or misbehave.**

**Dynamic memory allocation.**

* **The process of allocating memory during the time of execution is called dynamic memory allocation.**
* **Heap is the segment of the memory where dynamic memory allocation takes place.**
* **Unlike Stack where memory is allocated or deallocated in a defined order, heap is an area of memory where memory is allocated or deallocated without any order or randomly.**
* **There are some built in function which help in allocating or deallocating some memory space during run time.**
* **Pointers play a very important role in dynamic memory allocation.**
* **Allocated memory can only be accessed through pointers.**
* **The list of built in function that are used to allocate the memory during are :-**

**malloc()**

**calloc()**

**realloc()**

**free()**

**Dynamic Memory allocation using malloc():-**

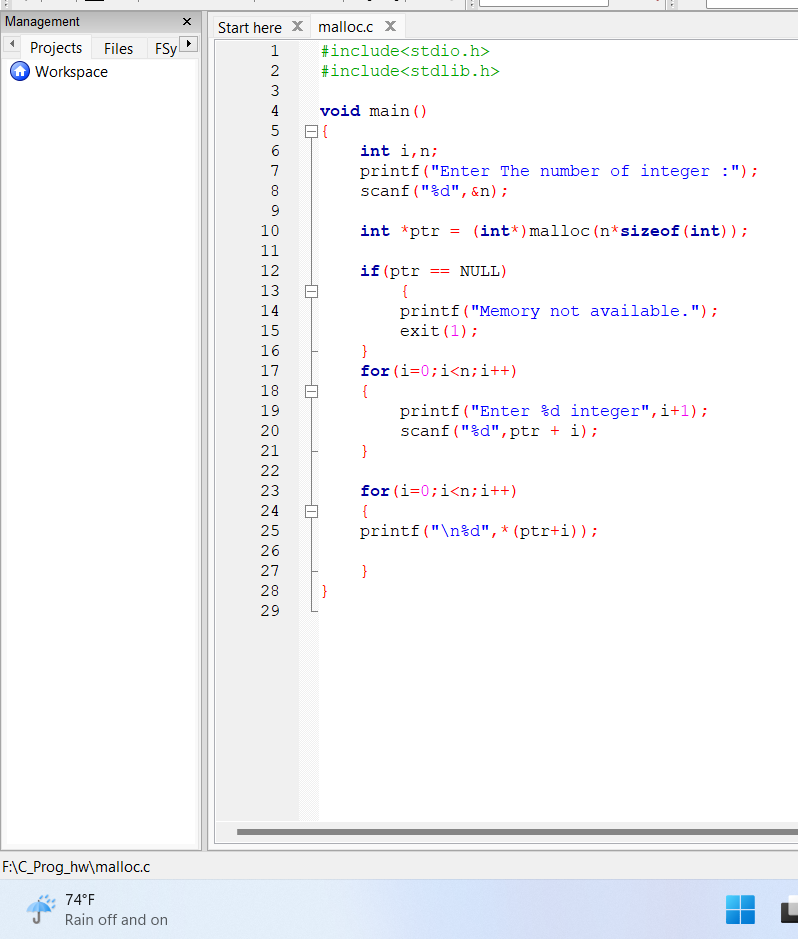
**What is malloc?**

* **Malloc is a built in function declared in the header file <stdlib.h>.**
* **Malloc is the short name of memory allocation and is used to dynamically allocate a single large block of contiguous memory according to the size specified.**
* **Syntax: - (void\*)malloc(size\_t size)**
* **Malloc function simply allocates a memory block according to the size specified in the heap and on success it returns a pointer pointing to the first byte of allocated memory else returns null.**
* **Size\_t is defined in <stdlib.h> as unsigned integer.**
* **Memory allocated by malloc is initialized with some garbage value.**

**Why malloc returns void pointer?**

* **Malloc doesn’t have an idea of what it is pointing to.**
* **It merely allocates memory requested by the user without knowing the type of data to be stored inside the memory.**
* **The void pointer can be type casted to an appropriate type.**
* **Syntax :- int \*ptr = (int\*)malloc(4).**

**Malloc allocates 4 bytes of memory in the heap and addresses of the first byte is stored in in the pointer ptr.**

****

**Dynamic memory allocation using calloc()**

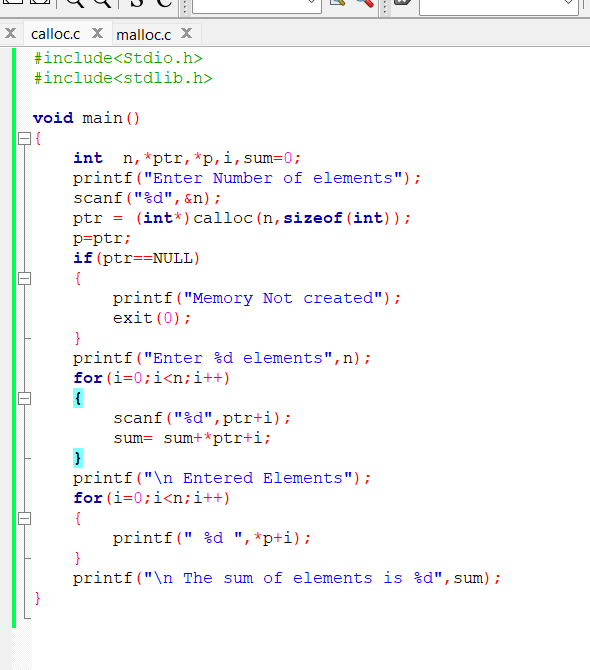
**What is calloc()?**

* **It stands for clear allocation.**
* **Calloc() function is used to dynamically allocate multiple blocks of memory.**
* **Calloc() is different from malloc in two ways:**
  + - **Calloc needs two arguments instead of just one.**
    - **Syntax: - void \*calloc (size\_t n, size\_t size)**

**1st argument represents number of blocks**

**2nd argument represent size of each block.**

* **Example: int \*ptr = (int\*)calloc(10,size of (int));**
* **An equivalent malloc call: int \*ptr = (int\*)malloc(10\*sizeof(int));** 
  + - **Memory allocated by calloc is initialized to zero**
* **It also returns the NULL when t sufficient memory is not available in the heap.**

****

**Dynamic memory allocation using realloc()**

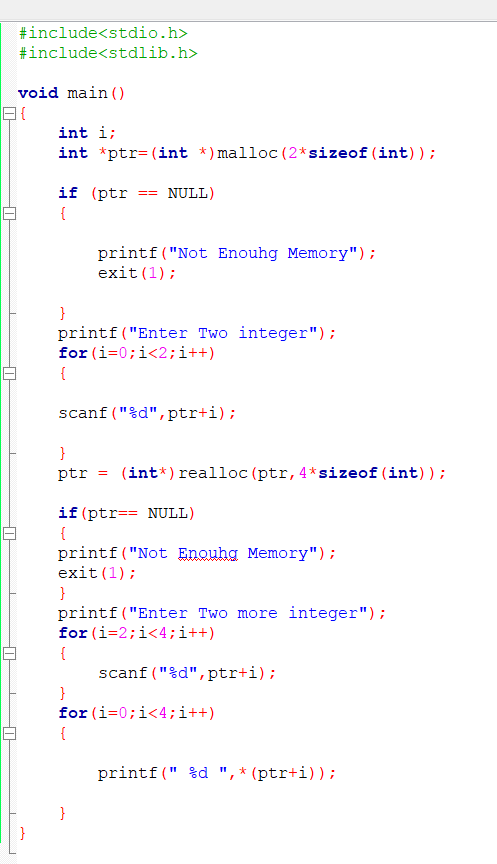
**What is realloc()?**

* **Realloc() is the function used to change the size of memory block without losing the old data.**
* **Syntax: - \*realloc(void \*ptr , size\_t new size)**
  + - **Where \*ptr is the pointer to previously allocated memory and new size is the new size of memory.**
* **Example :- \*ptr = (int\*)malloc(sizeof(int));**

**Ptr=(int\*)realloc(ptr,2\*sizeof(int));**

* **This function will allocate memory space of 2\*sizeof(int).**
* **Also this function will move the content of the old block to the new block and the data of the block is not lost.**
* **We may lost the data when the new size is smaller than the size of old size.**
* **Newly allocated bytes are uninitialized.**

**Program Example**

****

**Releasing the dynamically allocated memory Using free()**

**What is free()?**

* **free() function is used to release the dynamically allocated memory in the heap.**
* **Syntax: void free(ptr)**
* **The memory allocated in heap will not be released automatically after using the memory. The space remains there and and can’t be used.**
* **It is the programmer responsibility to release the memory after use.**
* **Example :-**

**int main()**

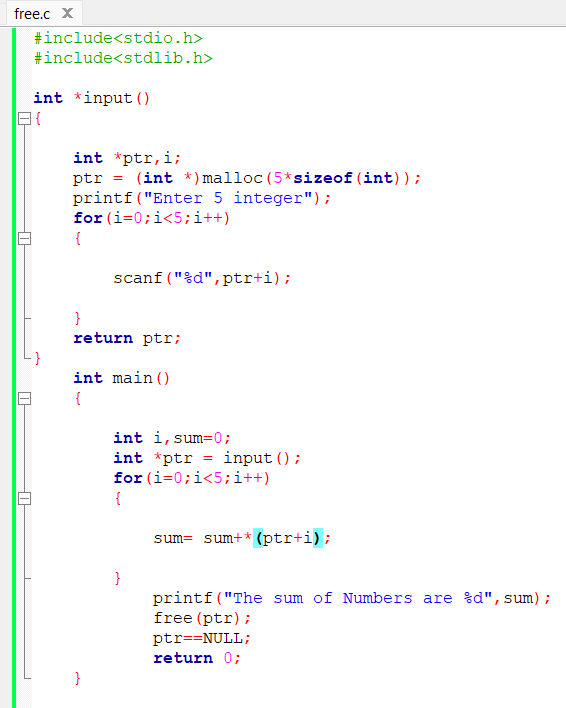
**{**

**int \*ptr =(int\*)malloc (4\*sizeof(int));**

**……**

**…….**

**free(ptr);**

**}**