M T W T F S S

DATE

Dynamic memory allocation

**C

→ Memory allocated during Compile time is Called Static memory.

↳ the memory allocated ~~during Compile time is~~ is fixed and cannot be increased or decreased during runtime.

```
int main() {
    int arr[5] = {1,2,3,4,5};
}
```

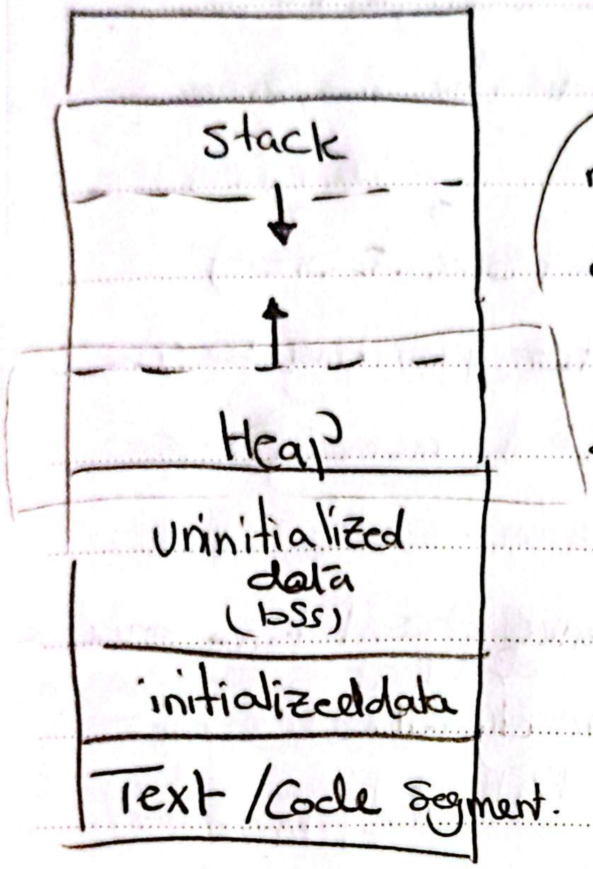
↑ memory allocated at Compile time is Fixed.

(Size is Fixed)

M T W T F S S
DATE

#C

→ The process of allocating memory at the time of excution is called [dynamic memory allocation]



(Heap) is the Segment of memory where dynamic memory allocation takes place.

→ memory is allocated or deallocated without any order or randomly.

unlike (Stack) memory is allocated or deallocated in defined order

[Allocated memory can only be accessed through pointers]

Built in functions:

malloc()

calloc()

realloc()

free()

M T W T F S S

*C

malloc() ⟶ header file <stdlib.h>

+"memory allocation"

used to dynamically allocate a single large block of contiguous memory according to the size specifie

unsigned int

Syntax (void*) malloc (size_t size)

it returns a pointer pointing to the first byte of the allocated memory else return NULL.

WHY void pointer ? → malloc doesn't have an idea of what is pointing to.
→ merely allocates memory without knowing type of data.
→ void pointer can be typecaste.

int *ptr = (int*) malloc(4)

↑ 4 bytes of memo

#C

```
#include <stdio.h>
# include <stdlib.h>

int main() {
   int i ,n;
   Printf ("Enter the number of integers:");
   Scanf ( "%d", &n);
  int *ptr =(int*) malloc(n*sizeof(int));
    if (ptr = = NUll) {
       Printf ("Memory not available.");
       exit (1); }
  For (i=0 ; i<n ; i++) {
    Printf ( "Enter aninteger : ");
    Scanf  ( "%d" , ptr +i ) ; }

  For( i =0, i<n, i++) {
     Printf ( "%d" , *(ptr+i));
     return 0;   }
```

M T W T F S S

※C

Calloc()
↳ Function is used to dynamically allocate multiple blocks of memory

Calloc() need two arguments instead just one

```
void *Calloc (size_t n , size_t size);
```

$$\begin{tikzcd} \text{size\_t n} & \text{size\_t size} \\ \text{no. of blocks} \arrow[u] & \text{Size of each block} \arrow[u] \end{tikzcd}$$

```
int *ptr = (int *)Calloc (10, Sizeof(int));
↳= int *ptr = (int*) malloc (10* Sizeof(int));
```

※ note :- Memory allocated by Calloc is initialized to Zero
Memory allocated by malloc is intialized with some garbage value

#C

realloc(); (reallocation.)

↳ function used to change the size of the memory block without losing the old data.

```
Void *realloc (void *ptr, Size_t newSize)
```

pointer to the previously allocated memory ↗ ↑ new sizer

→ On failure returns null.

→ This function moves the contents of the old block to anew block and the data of the old block is not lost.

→ We may lose the data when the new size is smaller than the old size.

#C

free();

↳ Function is used to release the dynamically allocated memory in heap.

Syntax: Void free(ptr)

*C

Preprocessor

↳ a preprocessor preprocess our program before it's being compiled

E.g: #include <Stdio.h>

↳ this #include preprocessor is used to include external header file in our program

define Preprocessor is used to define macros

#define PI 3.145

↑

macros

⟹ a macros is piece of Code which given a name in Program so where ever this text is written in the program the Compiler replace it with the value written so PI 3.145 ✓

PI $\boxed{=3.145}$ X

all of this is replaced ↗ (error)

M T W T F S S

DATE

List of preprocessor Directives

#define	used to define macros
#undef	used to undefine macros
# include	used to include external header file
# ifdef	used to include a Section of code if a Certain macro is defined by #defin
# ifndef	used to include a Section of Code if if Certain macro is not defined by # define.
# if	Check for Specfied Condition
#else	Alternate Code that excutes when #if false
# end if	Used to mark the end of #if, #ifdef and # ifndef
#error	Used to generate a Compilation error
# Line	used to modify line number and file name.

note :

```
#include "file_name"
#include <file_name>
```

File inclusion with double quotes (" ") tells the Compiler to Search for the header file in the directory of Source file while (< >) is used for System lib.

#C

#define preprocessor Directive: Macros

A macros is an idenfied in a #define preproccessor directive.
the macro-idenifier is replaced in the program with the replacement-text before the program ~~is~~ is Compiled.

Define Punction macros

```
# define Circle_Area(x) ((PI)*(x)*(x))

int main() {
    area = Circle_Area(4);
}
```

expanded to
is replaced by
((3.14159) * (4) * (4))

M T W T F S S
DATE

Macro Circle_Area Could be defined as a function

```
double CircleArea (double x)
 { return 3.14159 * x * x ;
 }
```

This functions performs the same Calculation ↳ maCro CIRcle_AREA
but "overhead of a function Call" is associated with function CircleArea.

#undef director