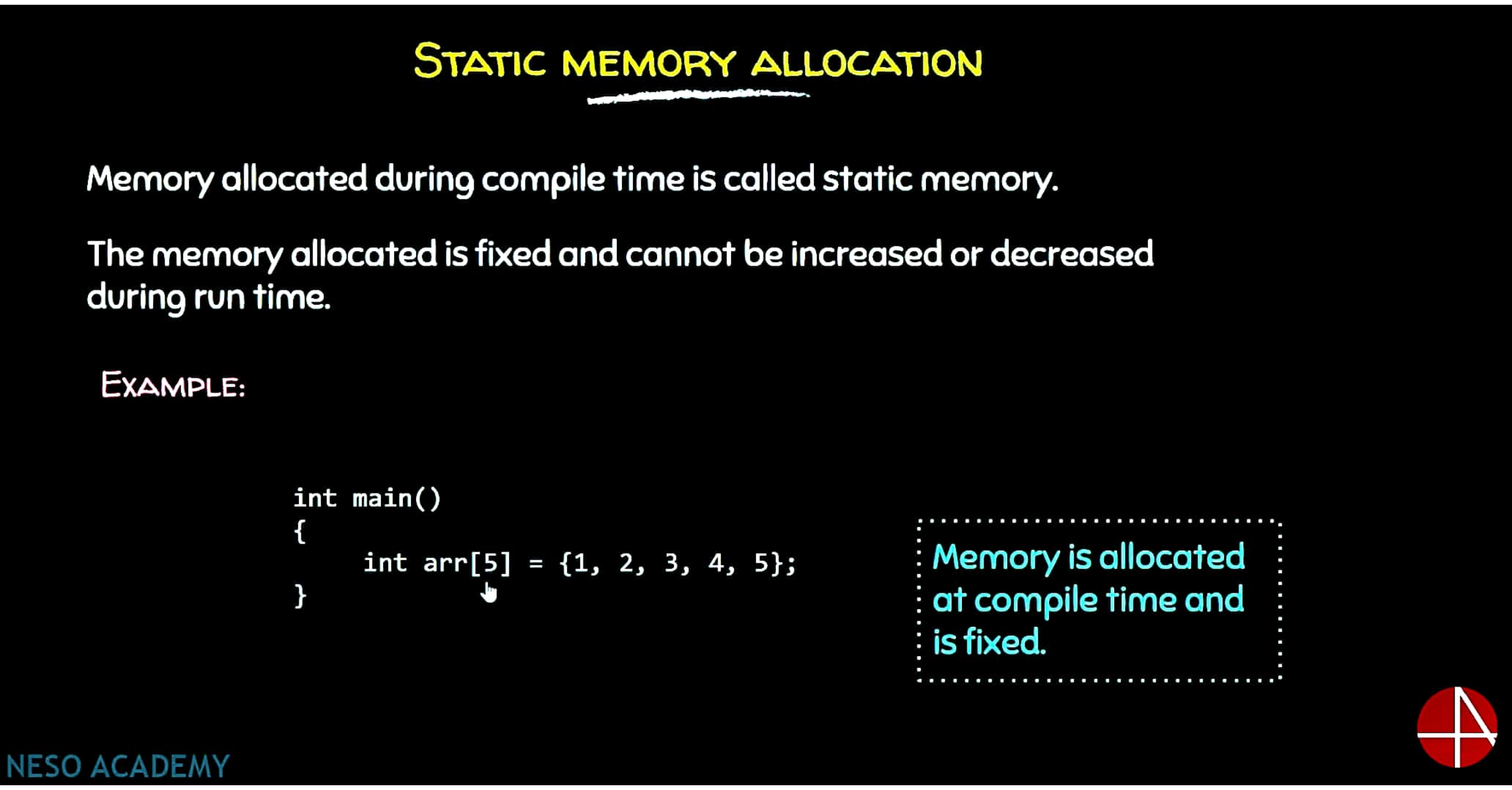


Static memory allocation

Memory allocated during compile time is called static memory.

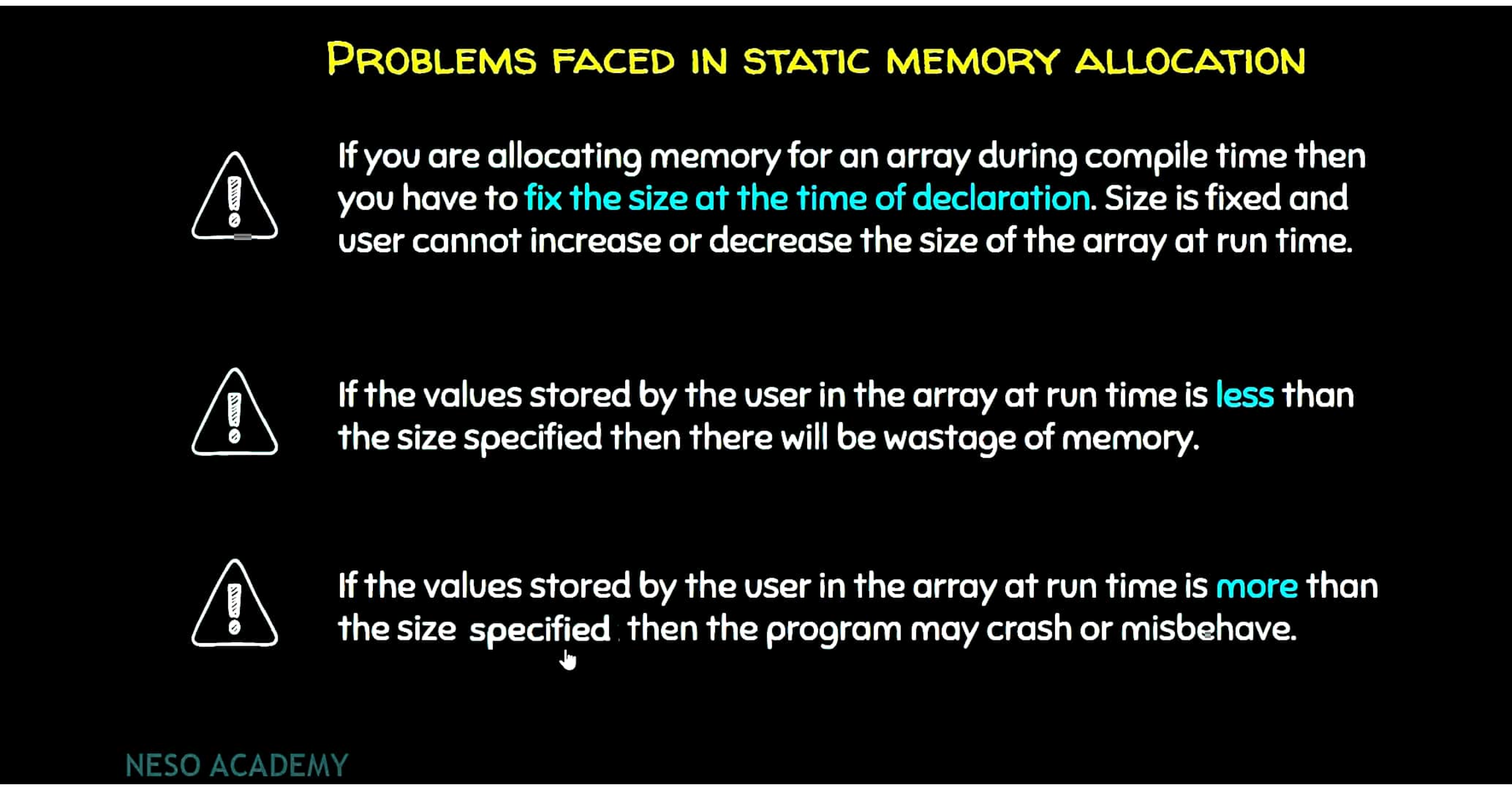
The memory allocated is fixed and cannot be increased or decreased during run time.

Example:

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

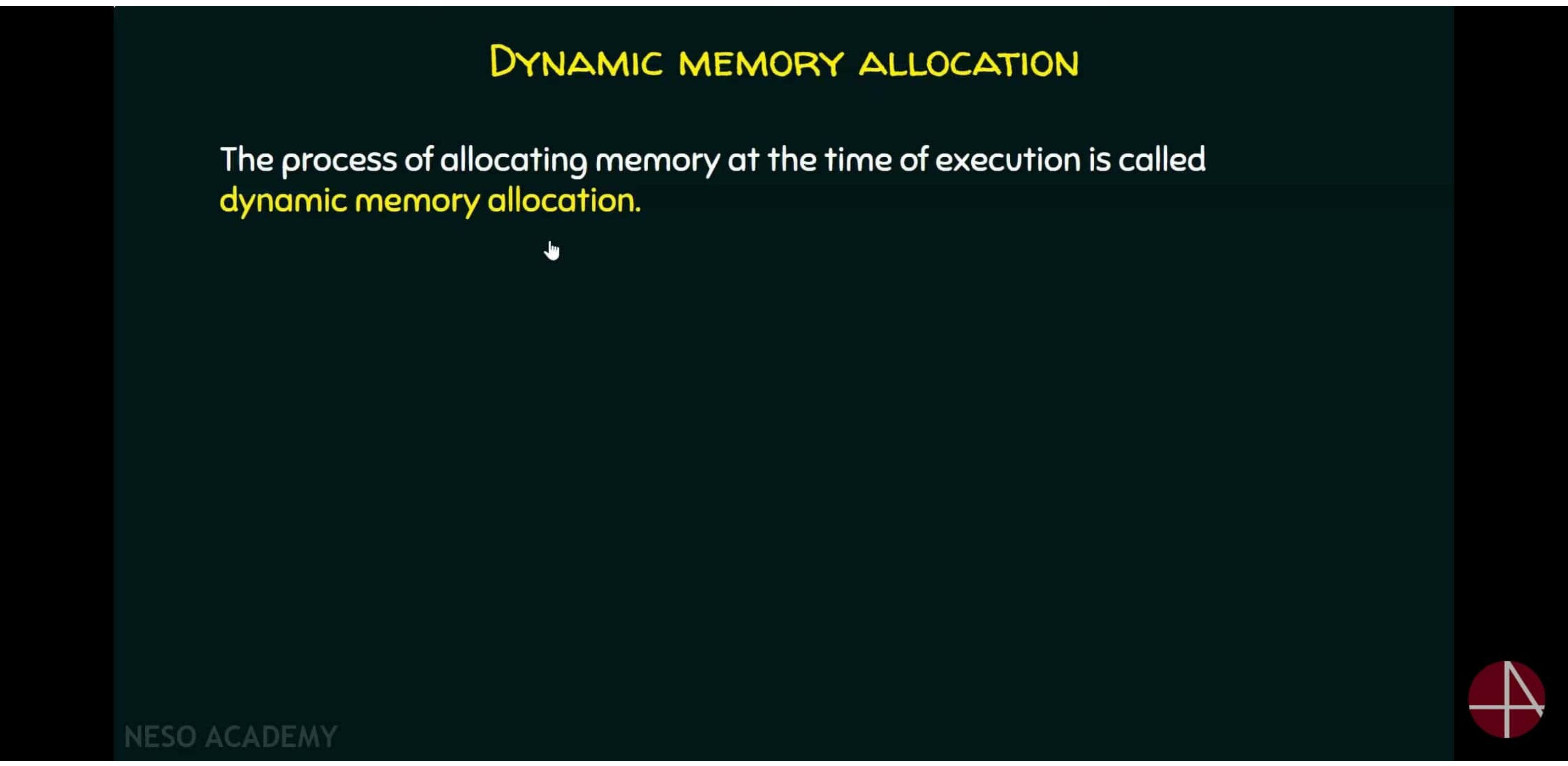
Memory is allocated at compile time and is fixed.

NESO ACADEMY



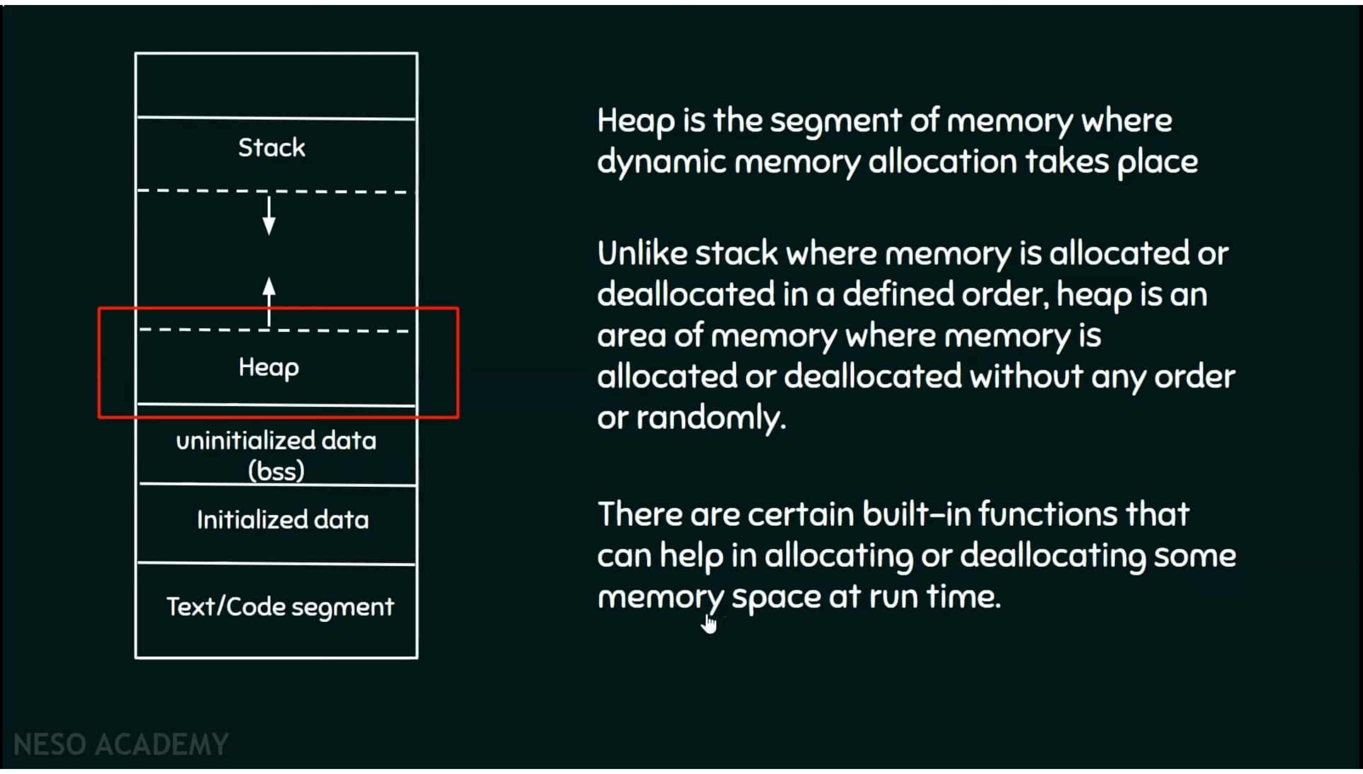
Problems faced in static memory allocation

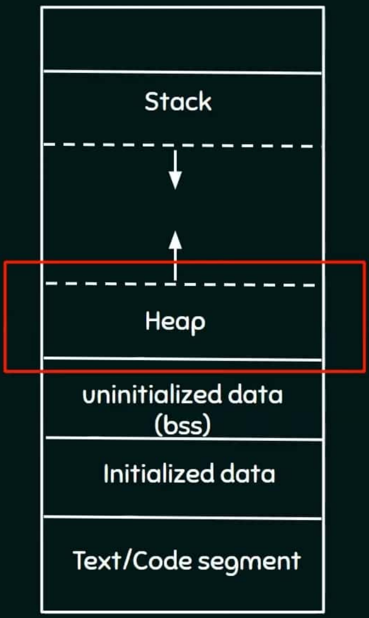
- If you are allocating memory for an array during compile time then you have to fix the size at the time of declaration. Size is fixed and user cannot increase or decrease the size of the array at run time.
- If the values stored by the user in the array at run time is less than the size specified then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



Dynamic memory allocation

The process of allocating memory at the time of execution is called dynamic memory allocation.

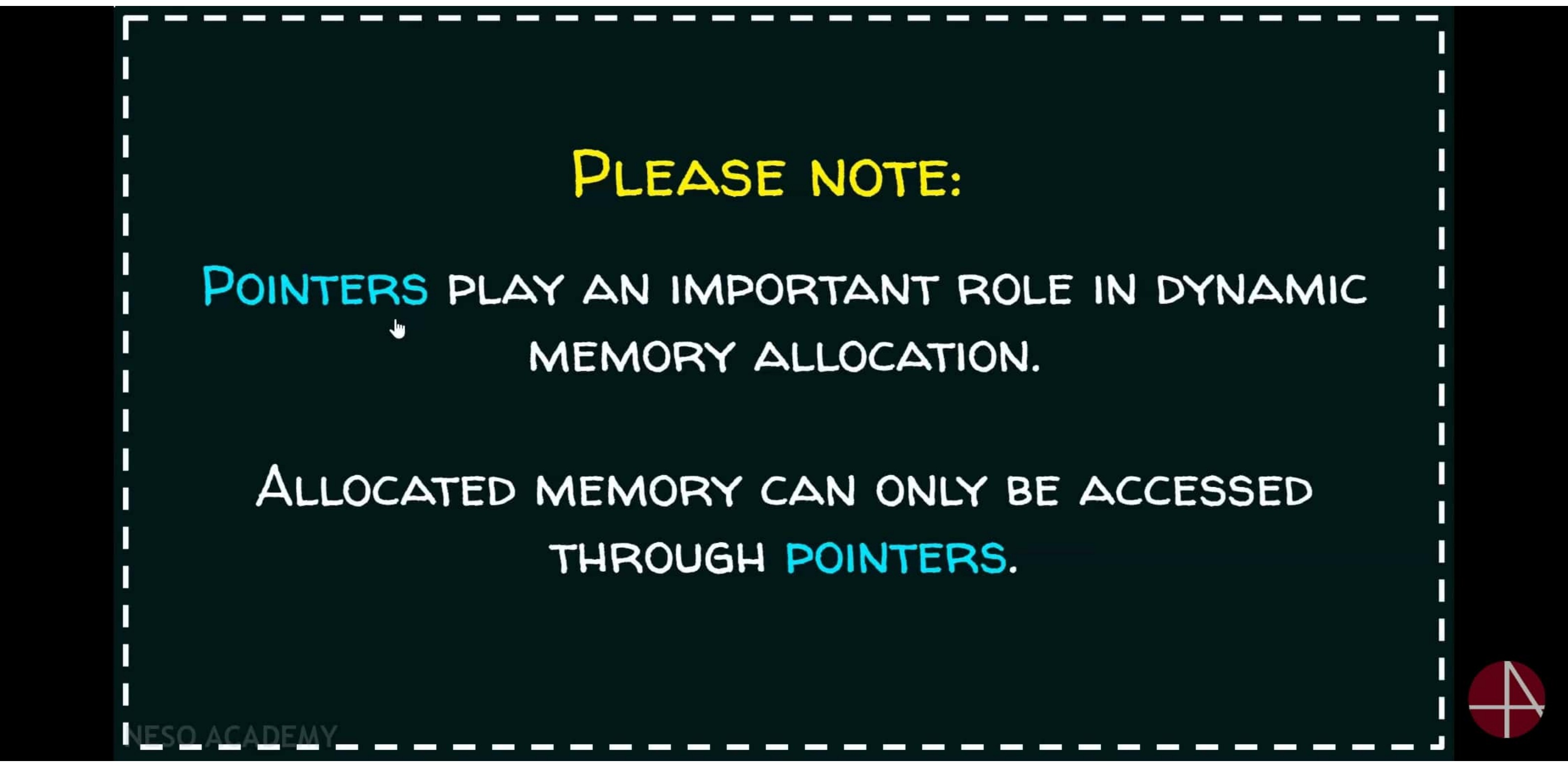




Heap is the segment of memory where dynamic memory allocation takes place

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.

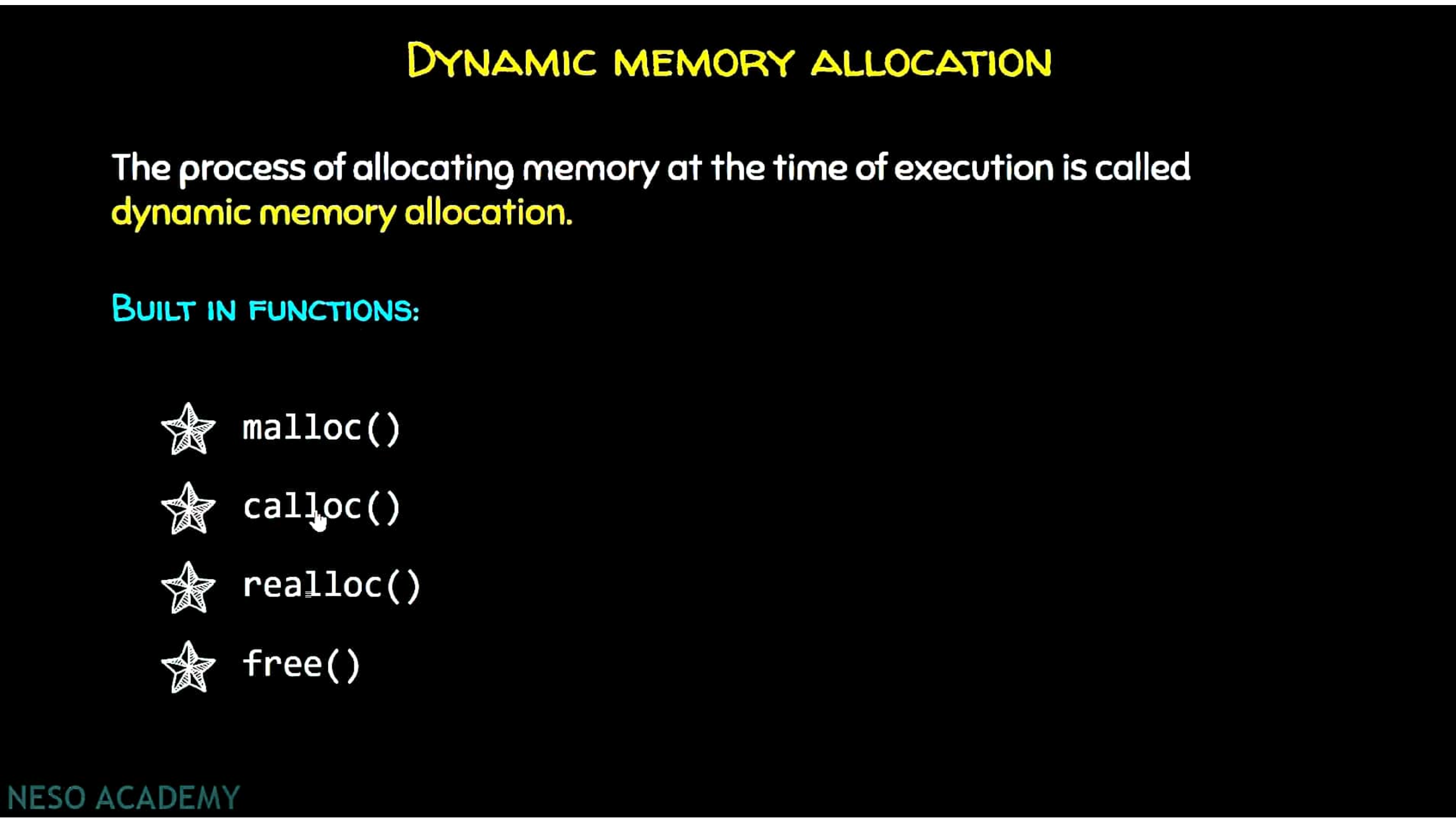
There are certain built-in functions that can help in allocating or deallocating some memory space at run time.



PLEASE NOTE:

POINTERS PLAY AN IMPORTANT ROLE IN DYNAMIC MEMORY ALLOCATION.

ALLOCATED MEMORY CAN ONLY BE ACCESSED THROUGH POINTERS.



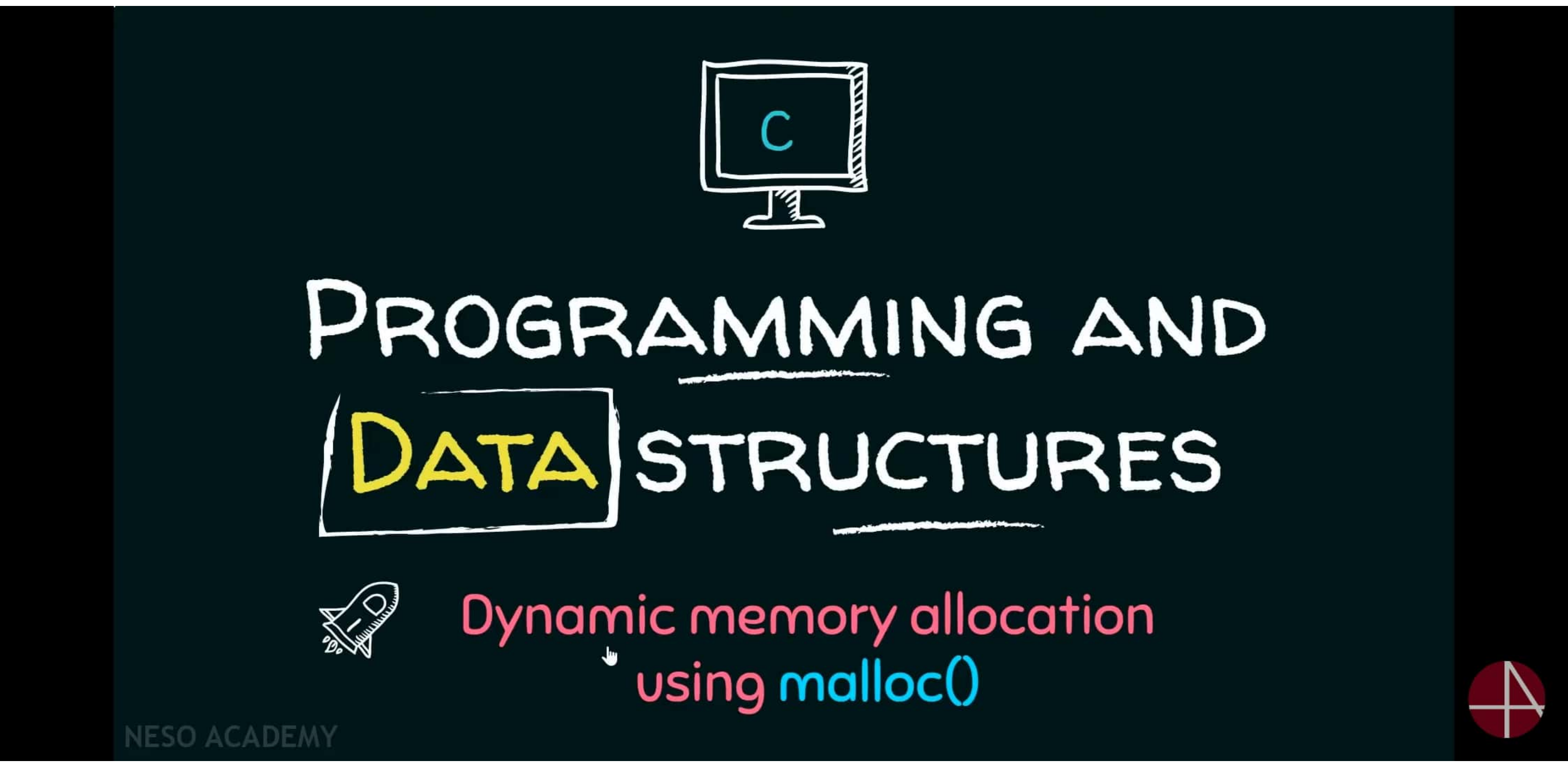
Dynamic memory allocation

The process of allocating memory at the time of execution is called **dynamic memory allocation.**

Built in functions:

- ☆ `malloc()`
- ☆ `calloc()`
- ☆ `realloc()`
- ☆ `free()`

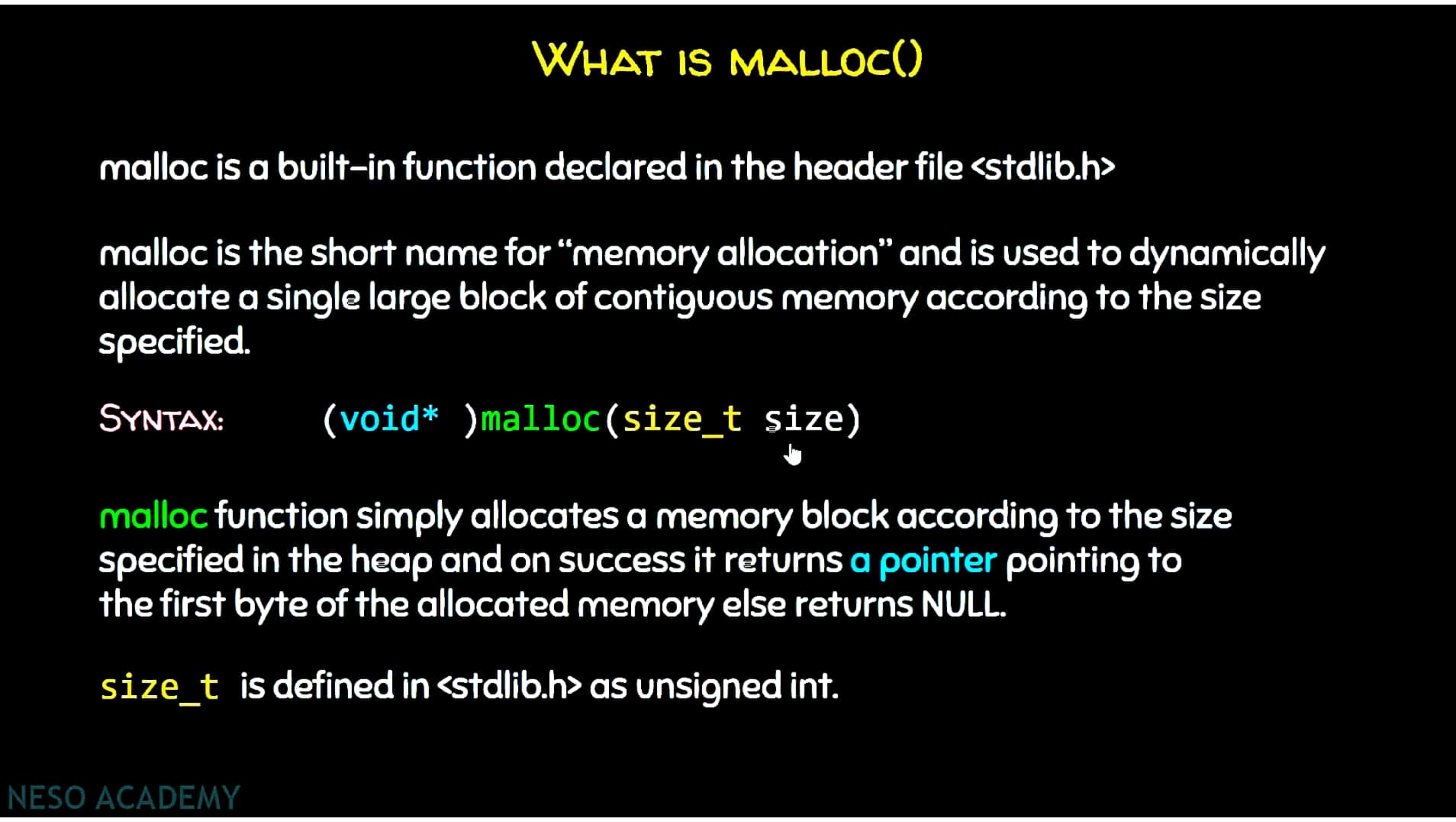
NESO ACADEMY



C

PROGRAMMING AND DATA STRUCTURES

Dynamic memory allocation using malloc()



WHAT IS MALLOC()

malloc is a built-in function declared in the header file <stdlib.h>

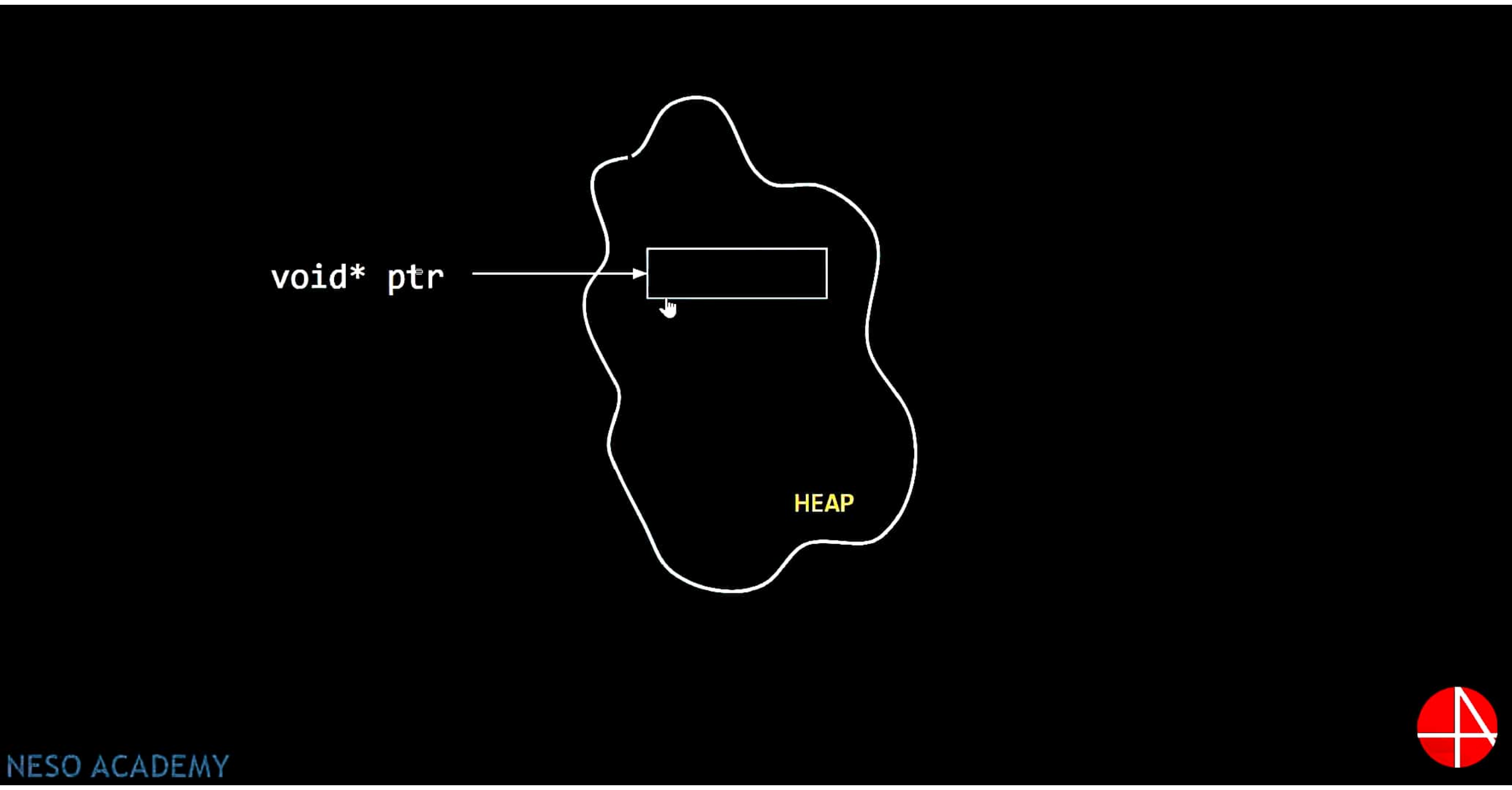
malloc is the short name for “memory allocation” and is used to dynamically allocate a single large block of contiguous memory according to the size specified.

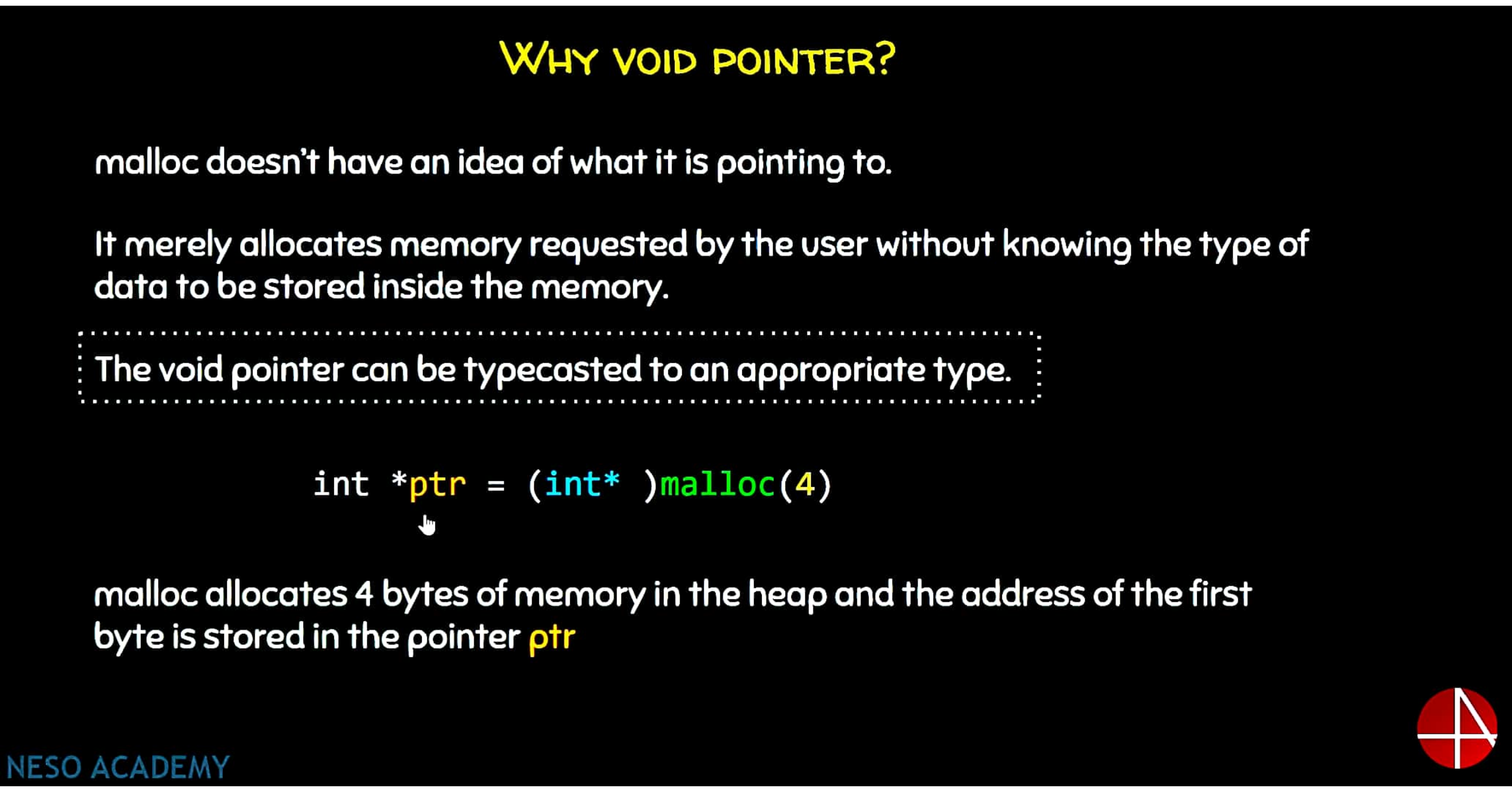
SYNTAX:

```
(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 the allocated memory else returns NULL.

`size_t` is defined in <stdlib.h> as unsigned int.


void* ptr
HEAP
NESO ACADEMY



WHY VOID POINTER?

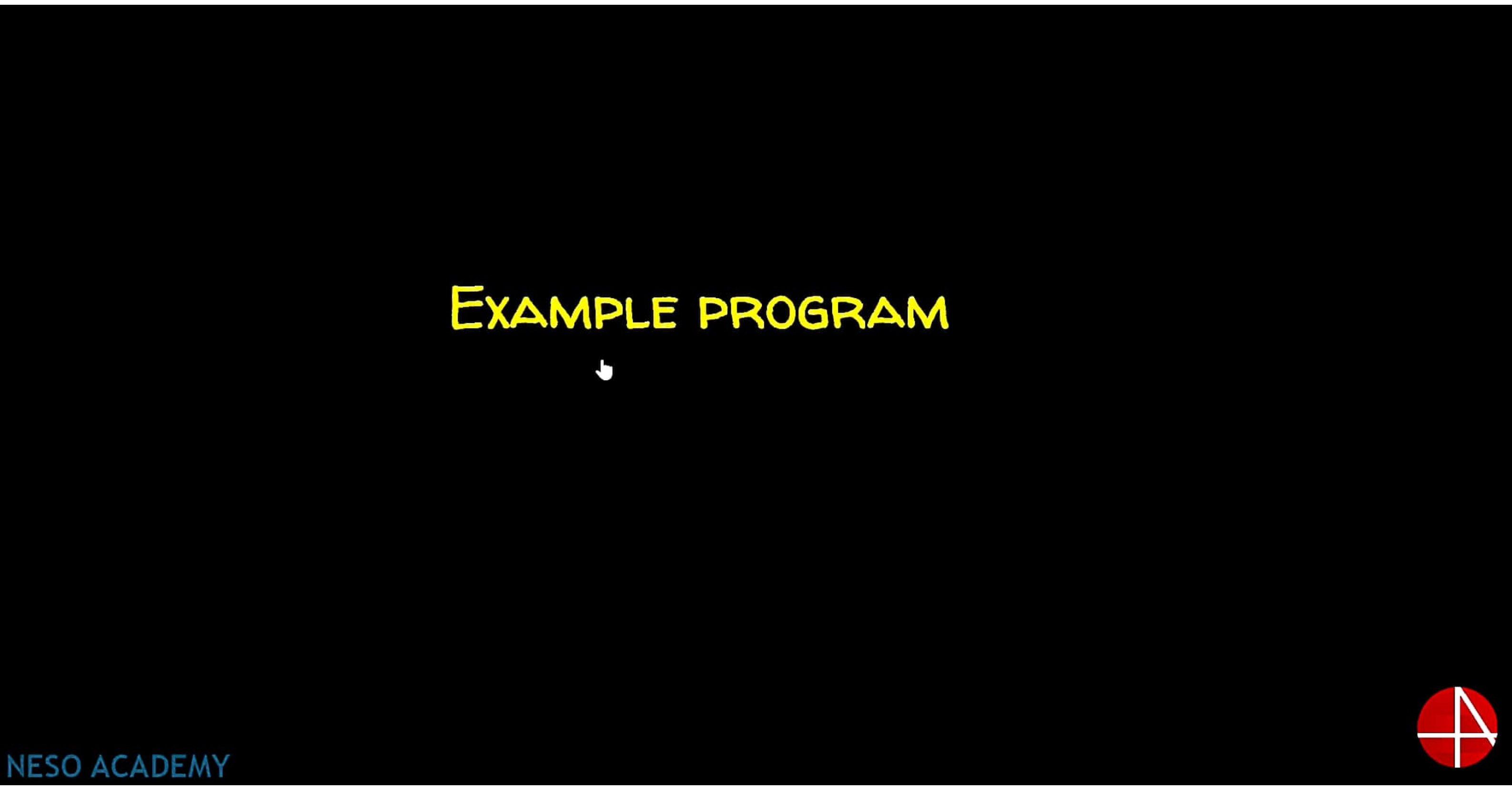
malloc doesn't have an idea of what it is pointing to.

It merely allocates memory requested by the user without knowing the type of data to be stored inside the memory.

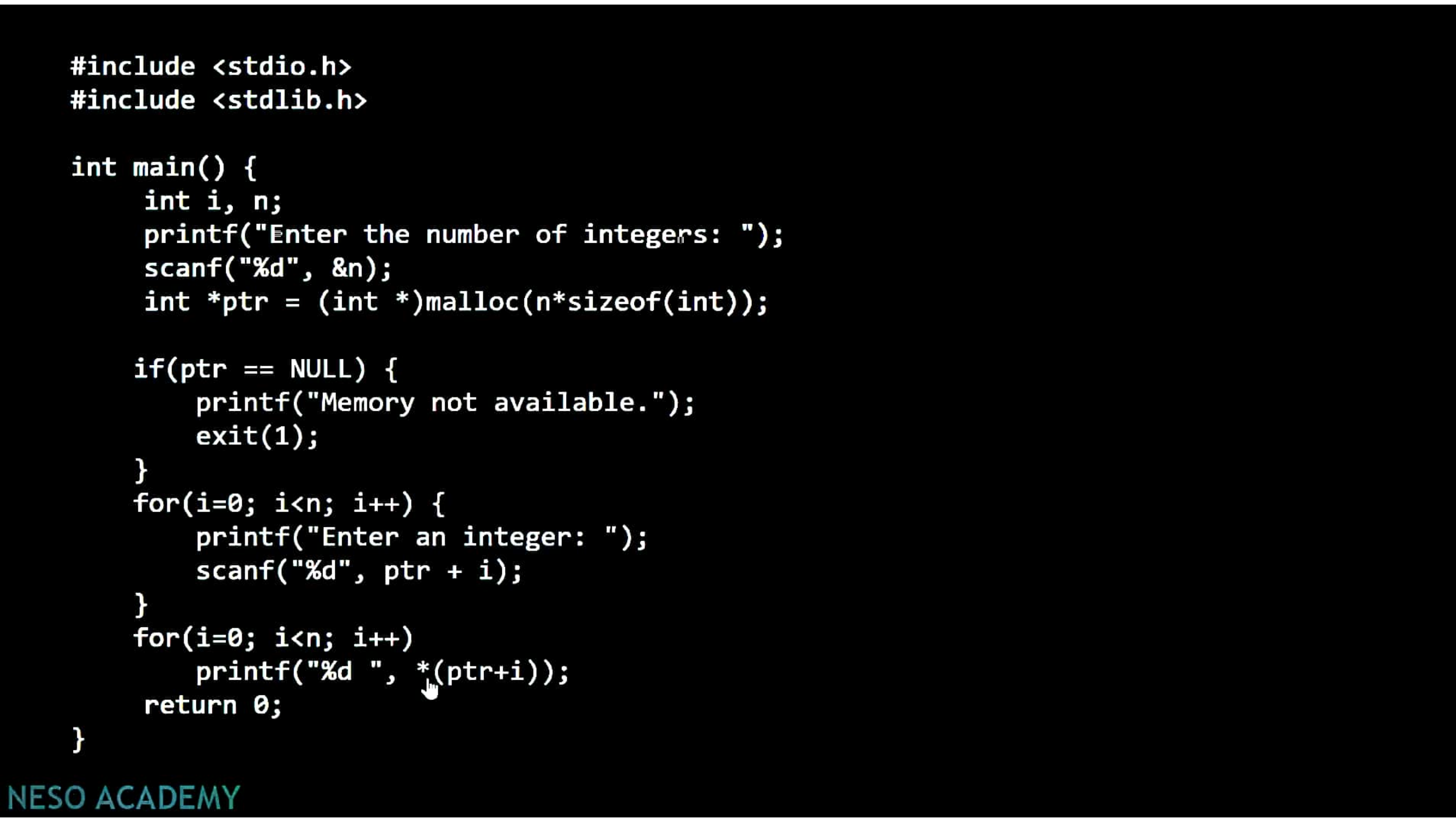
The void pointer can be typecasted to an appropriate type.

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

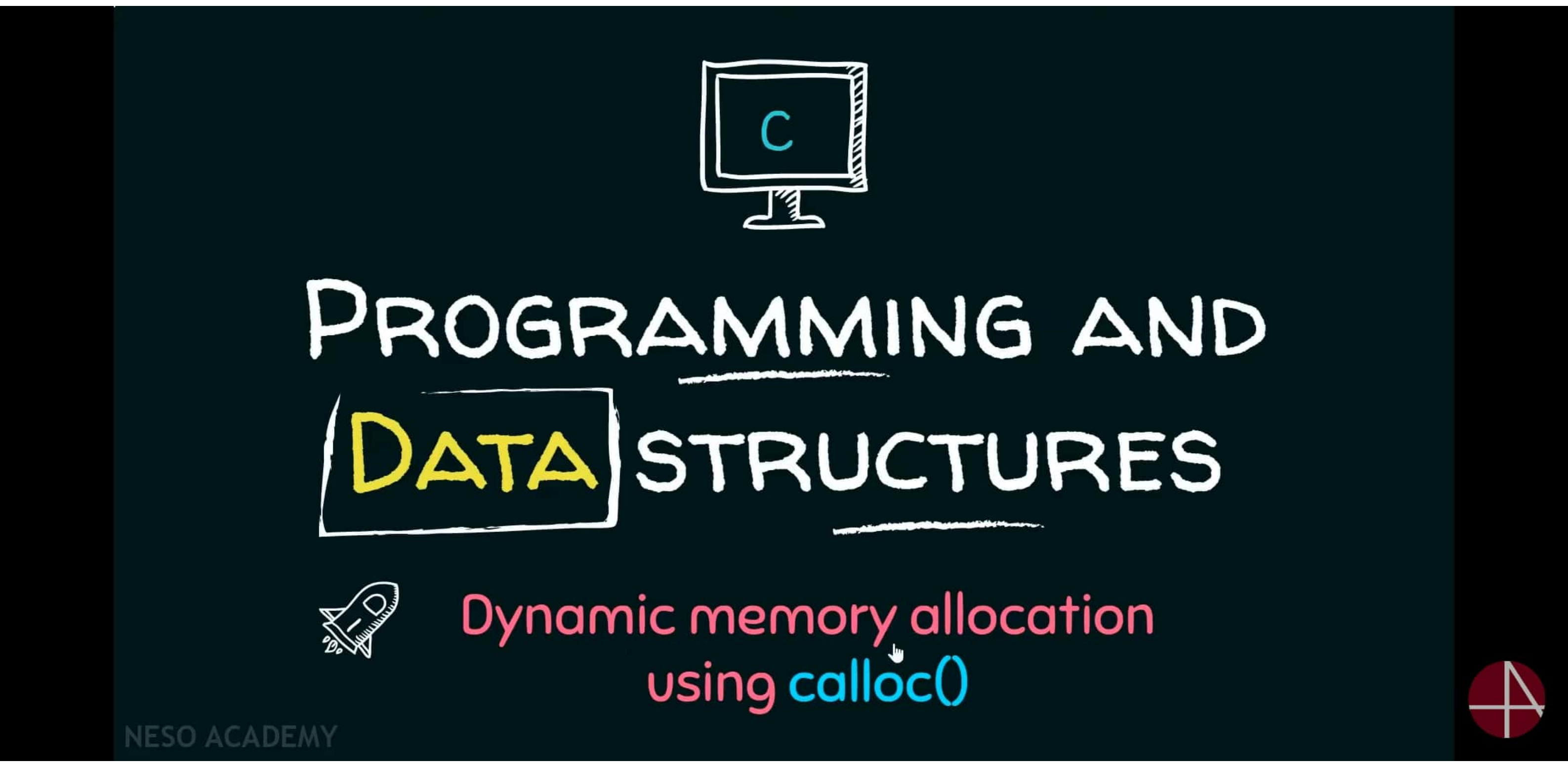
malloc allocates 4 bytes of memory in the heap and the address of the first byte is stored in the pointer ptr



Example program

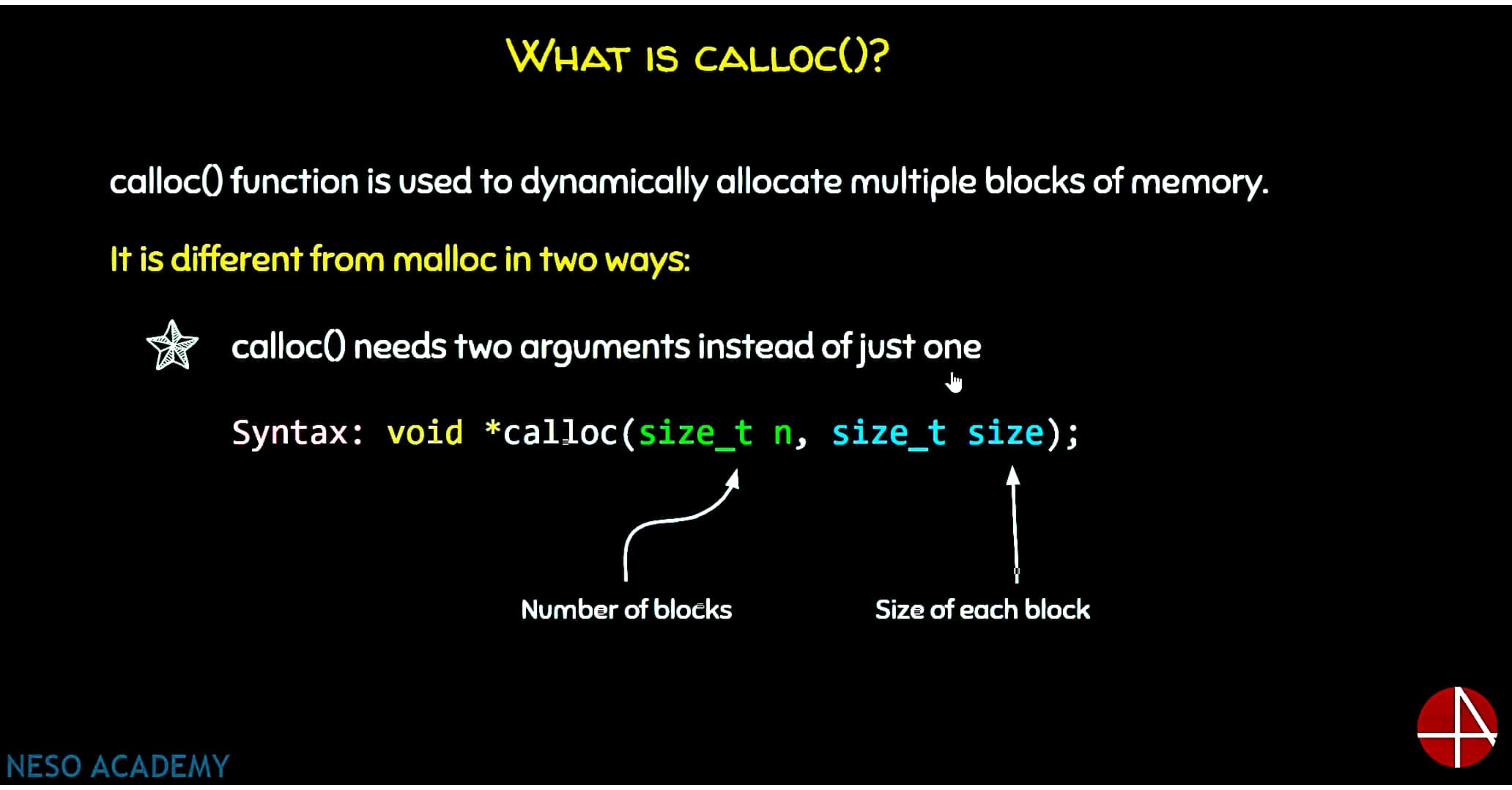


```
#include <stdio.h>
#include <stdlib.h>

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 an integer: ");
        scanf("%d", ptr + i);
    }
    for(i=0; i<n; i++)
        printf("%d ", *(ptr+i));
     return 0;
}
```



PROGRAMMING AND DATA STRUCTURES

Dynamic memory allocation using calloc()



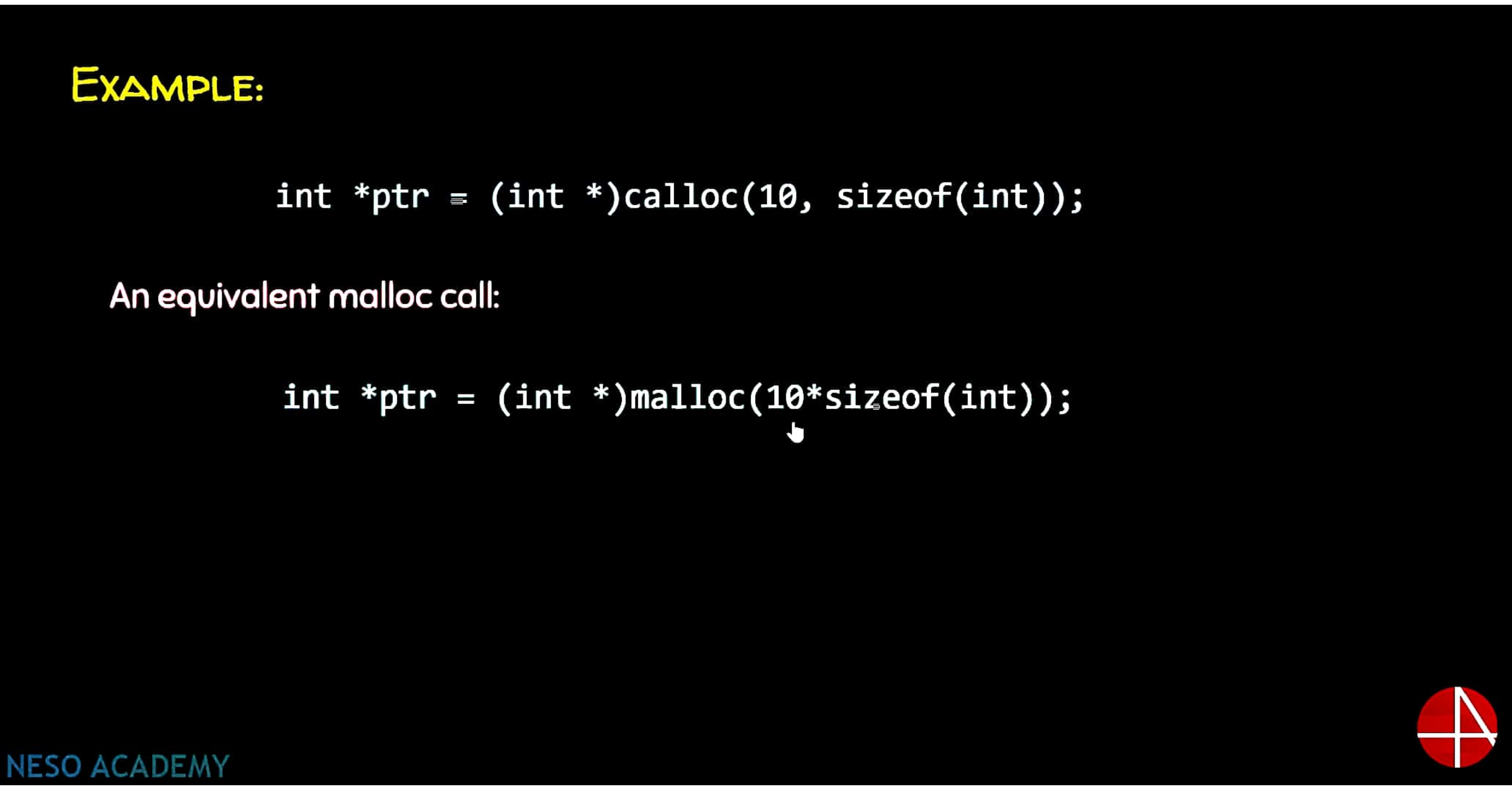
What is calloc()?

calloc() function is used to dynamically allocate multiple blocks of memory.

It is different from malloc in two ways:

- calloc() needs two arguments instead of just one

```
Syntax: void *calloc(size_t n, size_t size);
```



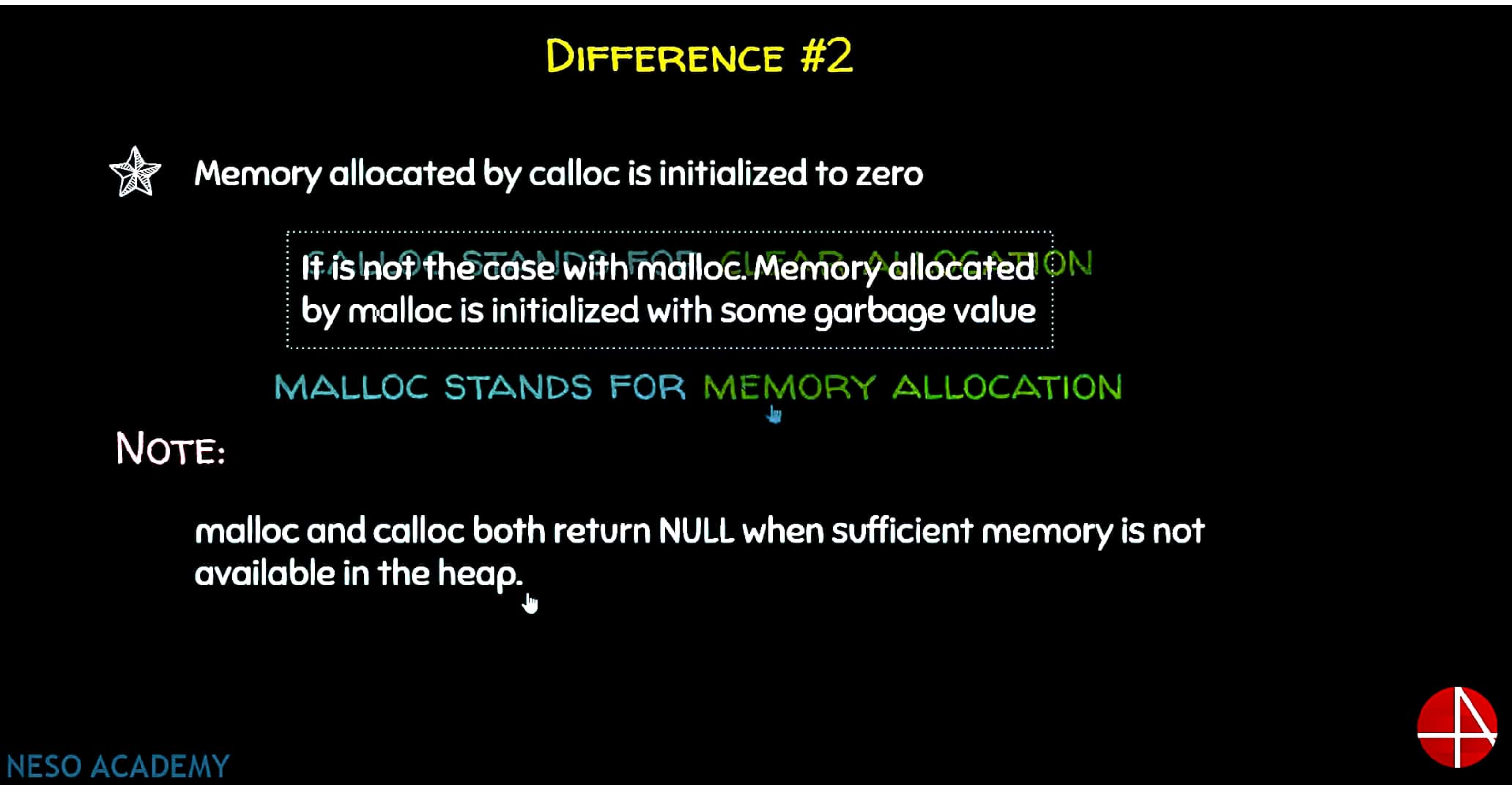
EXAMPLE:

```
int *ptr = (int *)calloc(10, sizeof(int));
```

An equivalent malloc call:

```
int *ptr = (int *)malloc(10*sizeof(int));
```

NESO ACADEMY



Difference #2

☆ Memory allocated by calloc is initialized to zero

It is not the case with malloc. Memory allocated by malloc is initialized with some garbage value

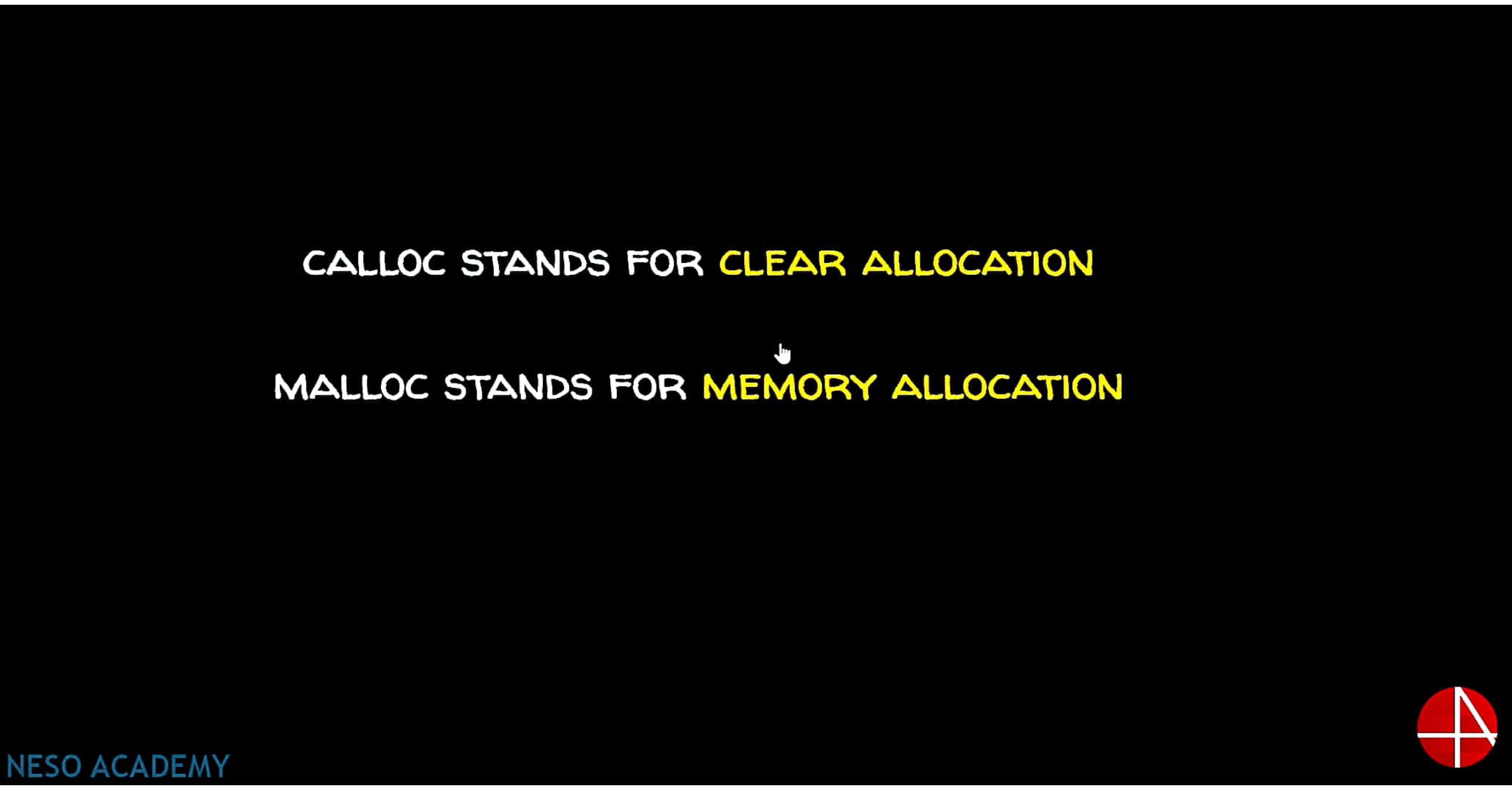
CALLOC STANDS FOR CLEAR ALLOCATION

MALLOC STANDS FOR MEMORY ALLOCATION

Note:

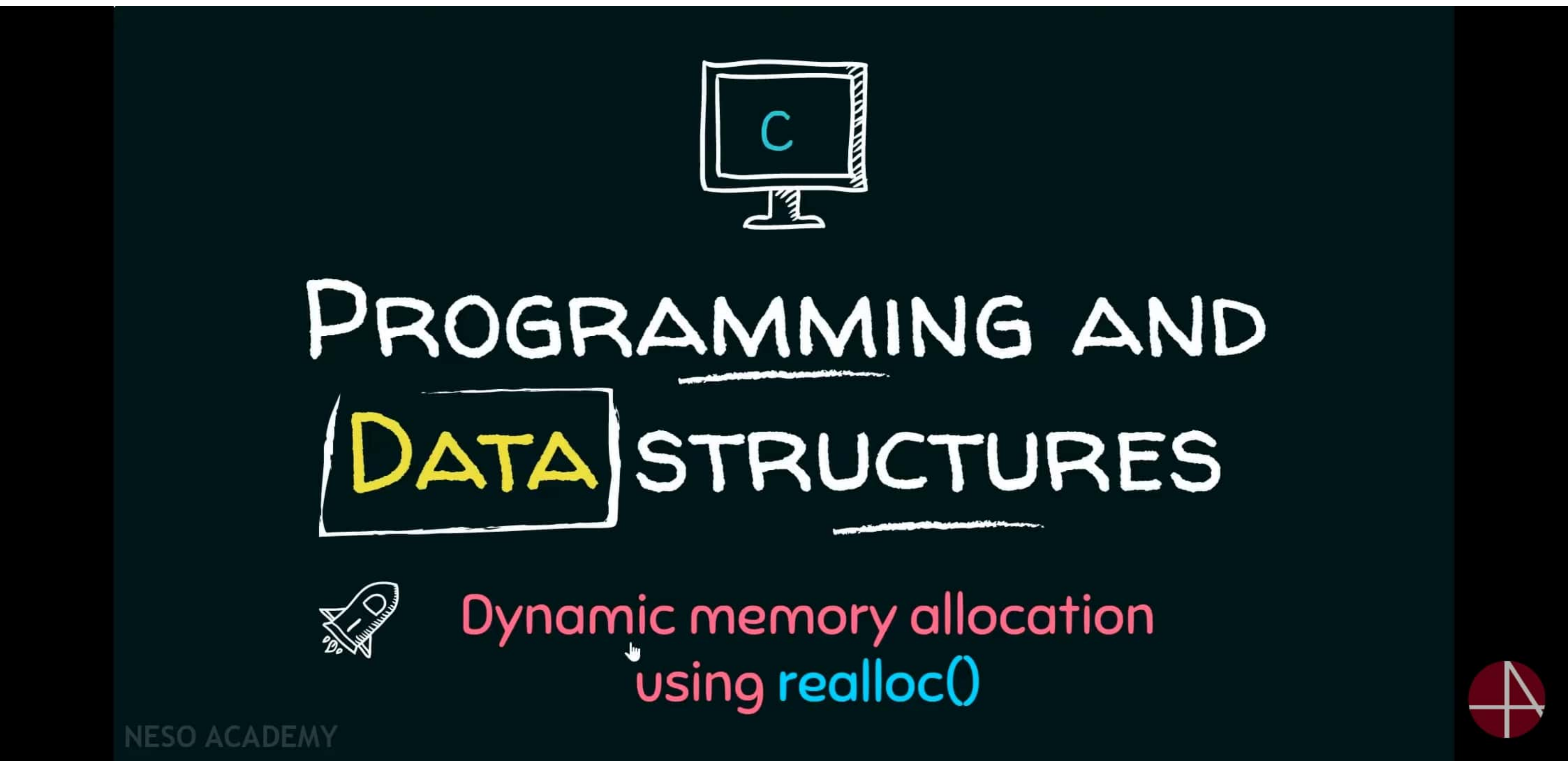
malloc and calloc both return NULL when sufficient memory is not available in the heap.

NESO ACADEMY



CALLOC STANDS FOR CLEAR ALLOCATION

MALLOC STANDS FOR MEMORY ALLOCATION



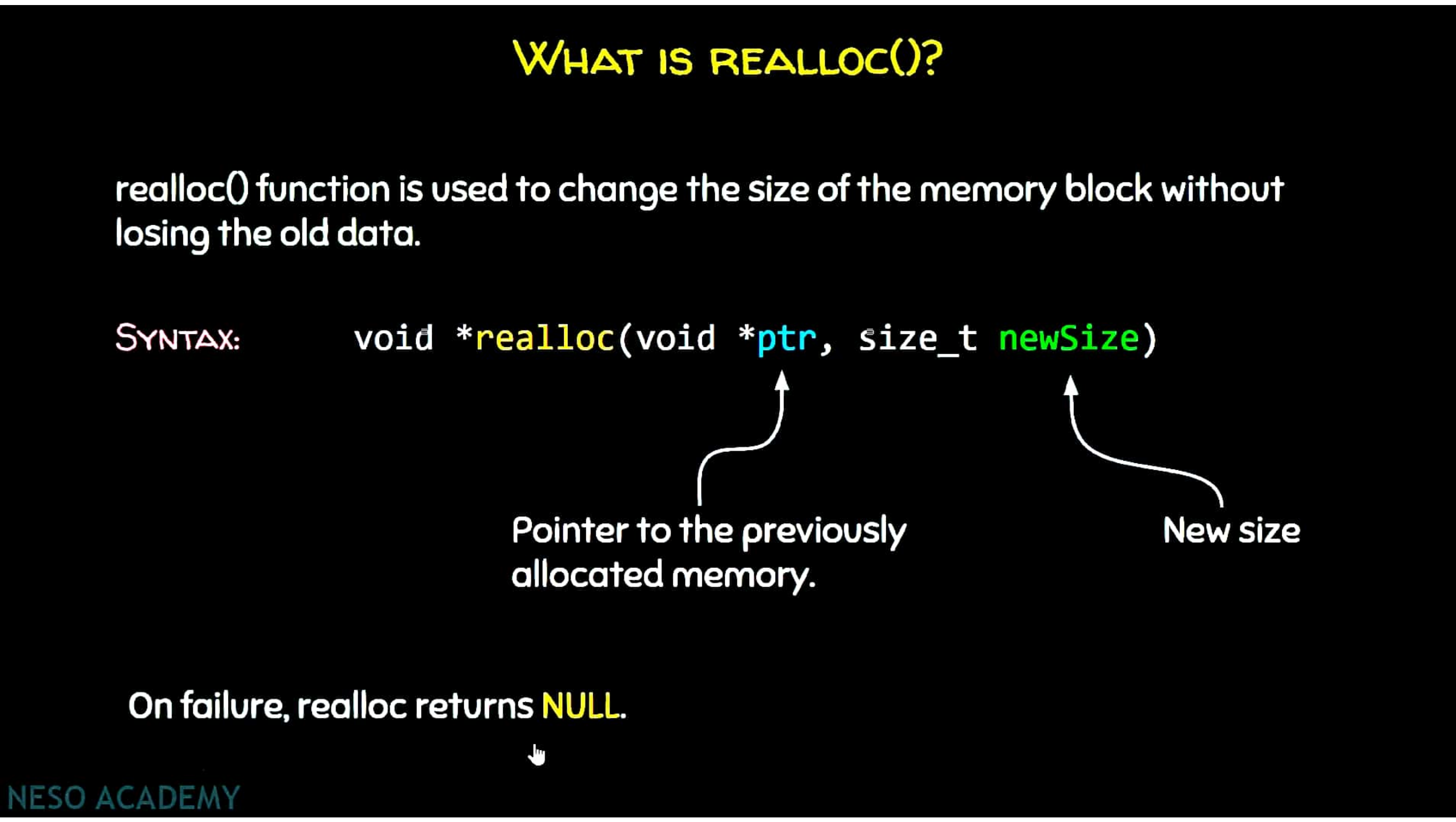
C

Programming and Data Structures

Dynamic memory allocation using realloc()

NESO ACADEMY

Scanned by CamScanner



What is realloc()?

realloc() function is used to change the size of the memory block without losing the old data.

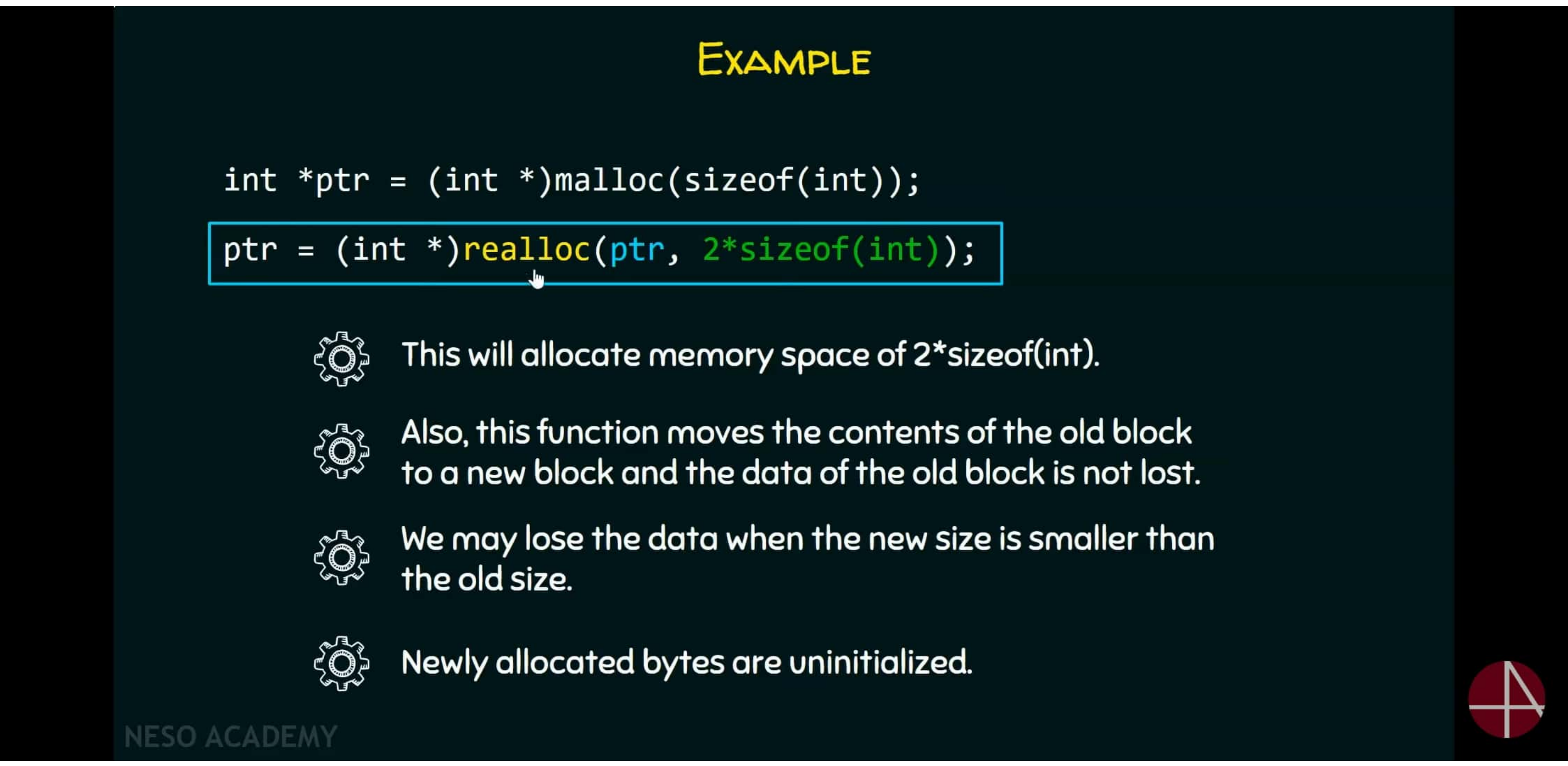
Syntax:

```
void *realloc(void *ptr, size_t newSize)
```

Pointer to the previously allocated memory.

New size

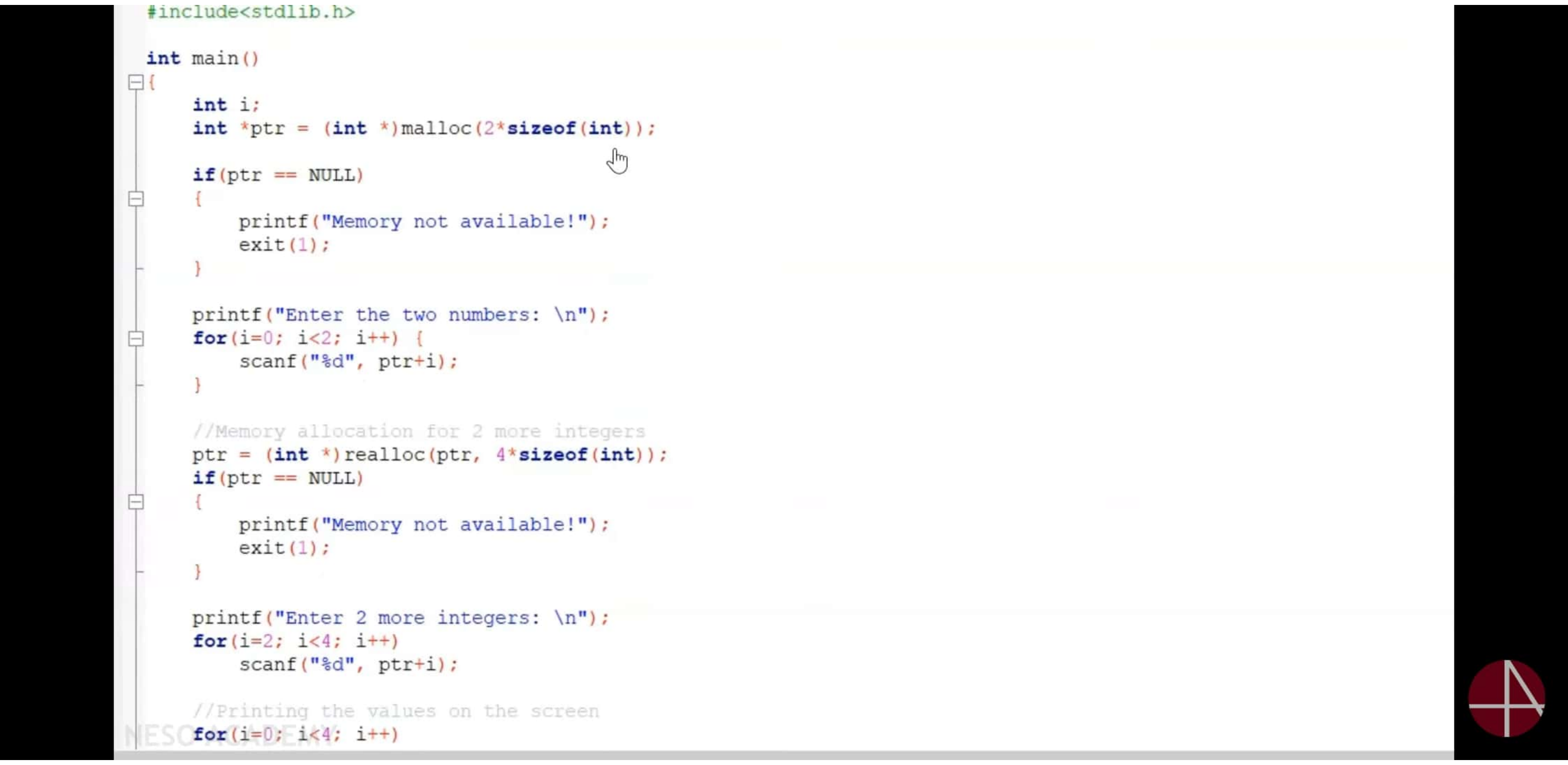
On failure, realloc returns NULL.



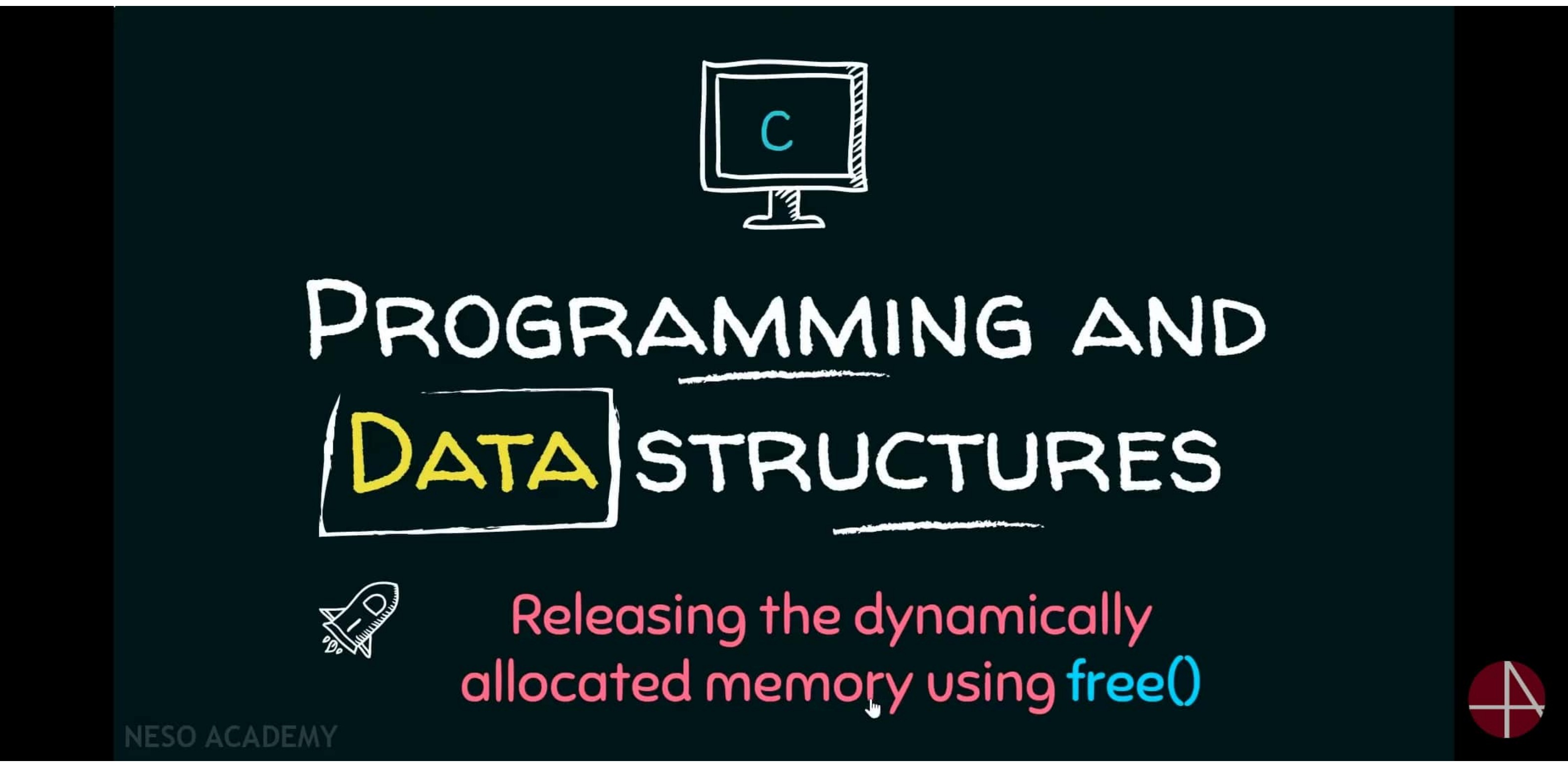
Example

```c
int *ptr = (int *)malloc(sizeof(int));
ptr = (int *)realloc(ptr, 2*sizeof(int));
```

- This will allocate memory space of 2*sizeof(int).
- Also, this function moves the contents of the old block to a new 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.
- Newly allocated bytes are uninitialized.



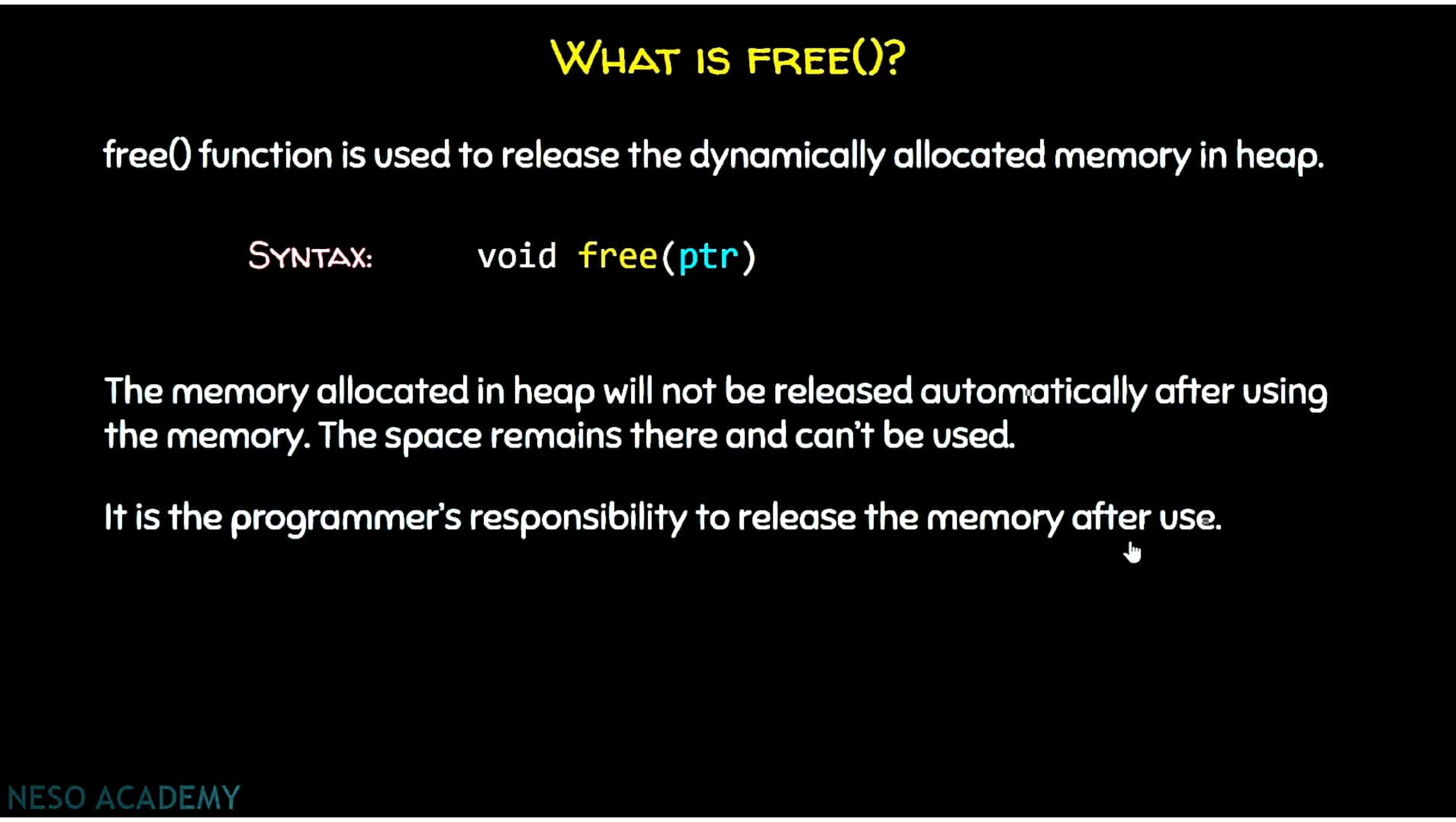
```c
#include<stdlib.h>

int main()
{
    int i;
    int *ptr = (int *)malloc(2*sizeof(int));

    if(ptr == NULL)
    {
        printf("Memory not available!");
        exit(1);
    }

    printf("Enter the two numbers: \n");
    for(i=0; i<2; i++) {
        scanf("%d", ptr+i);
    }

    //Memory allocation for 2 more integers
    ptr = (int *)realloc(ptr, 4*sizeof(int));
    if(ptr == NULL)
    {
        printf("Memory not available!");
        exit(1);
    }

    printf("Enter 2 more integers: \n");
    for(i=2; i<4; i++)
        scanf("%d", ptr+i);

    //Printing the values on the screen
    for(i=0; i<4; i++)
```



PROGRAMMING AND DATA STRUCTURES

Releasing the dynamically allocated memory using free()

NESO ACADEMY



WHAT IS FREE()?

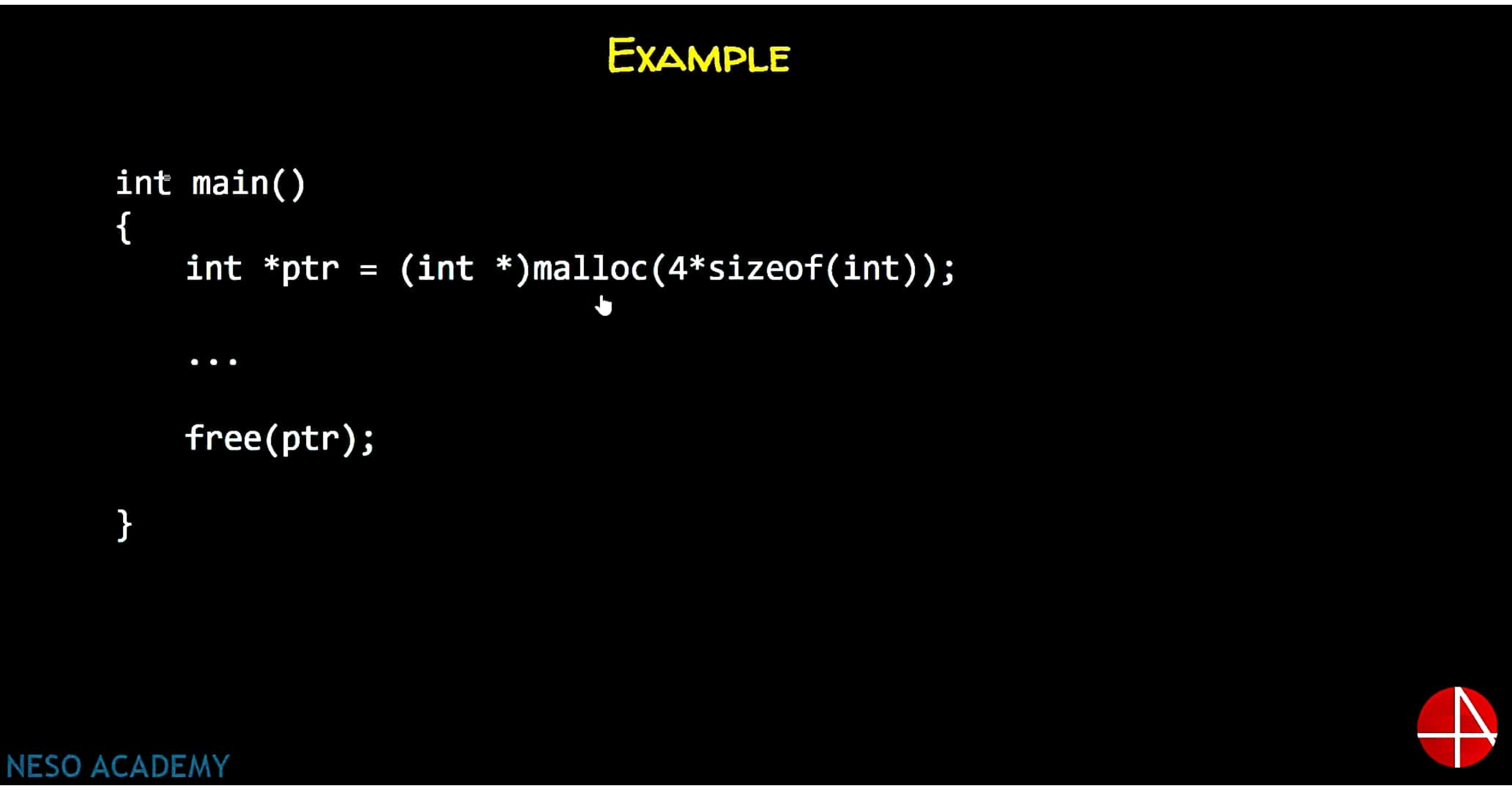
free() function is used to release the dynamically allocated memory in heap.

SYNTAX:

```
void free(ptr)
```

The memory allocated in heap will not be released automatically after using the memory. The space remains there and can't be used.

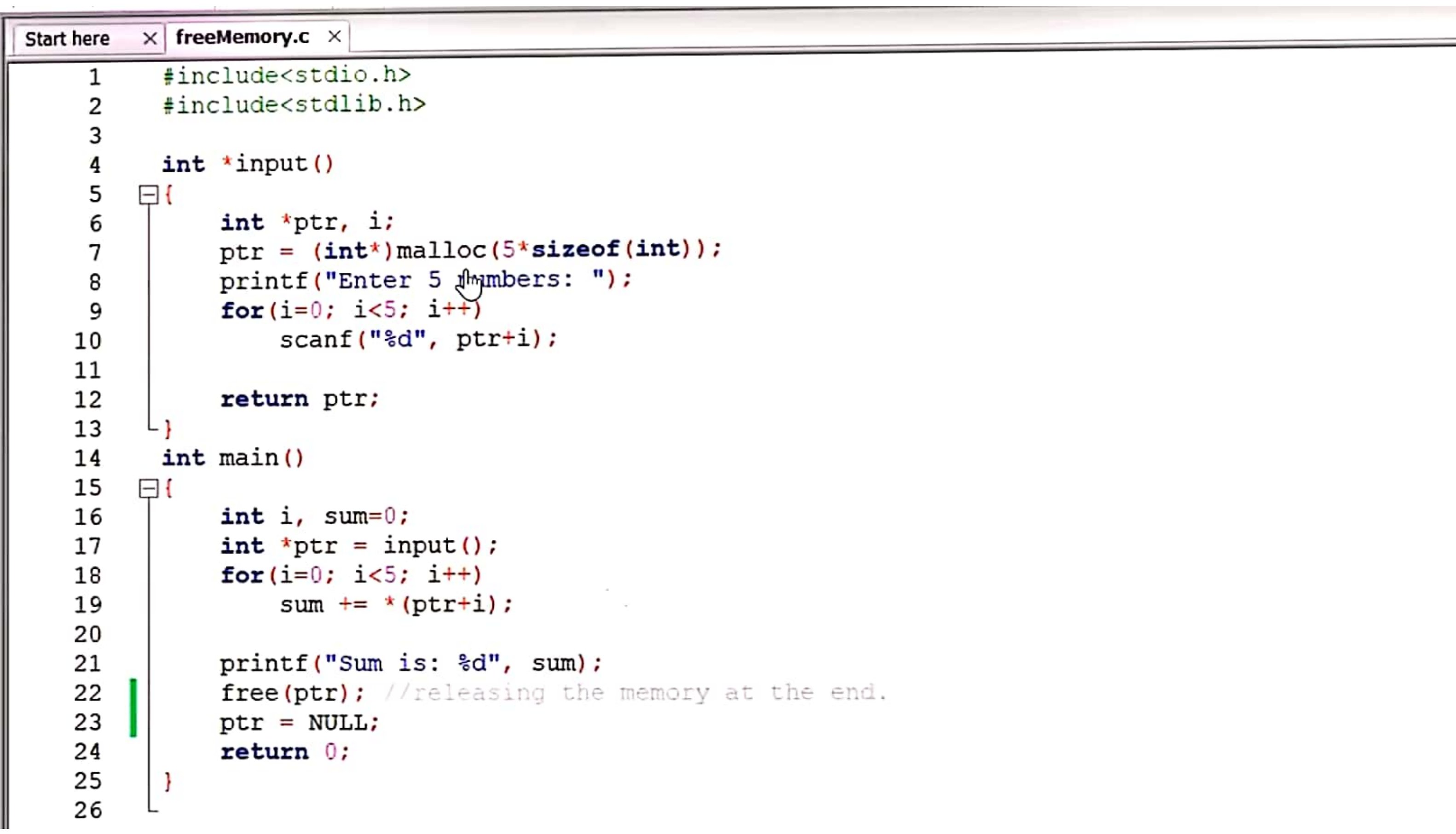
It is the programmer's responsibility to release the memory after use.



EXAMPLE

```c
int main()
{
    int *ptr = (int *)malloc(4*sizeof(int));

    ...

    free(ptr);

}
```

NESO ACADEMY



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

int *input()
{
    int *ptr, i;
    ptr = (int*)malloc(5*sizeof(int));
    printf("Enter 5 numbers: ");
    for(i=0; i<5; i++)
        scanf("%d", ptr+i);

    return ptr;
}
int main()
{
    int i, sum=0;
    int *ptr = input();
    for(i=0; i<5; i++)
        sum += *(ptr+i);

    printf("Sum is: %d", sum);
    free(ptr); //releasing the memory at the end.
    ptr = NULL;
    return 0;
}
```

Chapter 11 - Practice Set

1. Write a program to dynamically create an array of size 6 capable of storing 6 integers.

2. Use the array in problem 1 to store 6 integers entered by the user

3. Solve problem 1 using calloc()

4. Create an array dynamically capable of storing 5 integers. Now use realloc so that it can now store 10 integers.

5. Create an array of multiplication table of 7 upto 10 (7×10 = 70). Use realloc to make it store 15 numbers (from 7×1 to 7×15).

6. Attempt problem 4 using calloc().

Chapter 11 - Dynamic Memory Allocation

C is a language with some fixed rules of programming. For example: changing the size of an array is not allowed.

Dynamic Memory Allocation

Dynamic memory allocation is a way to allocate memory to a data structure during the runtime. We can use DMA functions available in C to allocate and free memory during runtime.

Functions for DMA in C

Following functions are available in C to perform Dynamic memory Allocation:

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

malloc() function

malloc stands for memory allocation. It takes number of bytes to be allocated as an input and returns a pointer of type void

Syntax:

```
ptr = (int*) malloc(30* sizeof(int))
```

→ returns size of 1 int

casting void pointer to int

space for 30 ints

The expression returns a null pointer if the memory cannot be allocated.

Quick Quiz: Write a program to create a dynamic array of 5 floats using malloc().

Calloc() function
Calloc stands for continuous allocation.
It initializes each memory block with a default value of 0.

Syntax:

```
ptr = (float*) calloc(30, sizeof(float));
```

↓ Allocates contiguous space in memory for 30 blocks (floats)

If the space is not sufficient, memory allocation fails and a NULL pointer is returned.

Quick Quiz: Write a program to create an array of size n using calloc where n is an integer entered by the user.

free() function
We can use free() function to de allocate the memory.
The memory allocated using calloc/malloc is not deallocated automatically.

Syntax :

```
free(ptr);
```
⟹ Memory of ptr is released.

Quick Quiz : Write a program to demonstrate the usage of free() with malloc().

realloc() function
Sometimes the dynamically allocated memory is insufficient or more than required.

realloc is used to allocate memory of new size using the previous pointer and size.

Syntax :

```
ptr = realloc(ptr, newSize);
ptr = realloc(ptr, 3* sizeof(int));
```
⇓
ptr now points to this new block of memory capable of storing 3 integers.