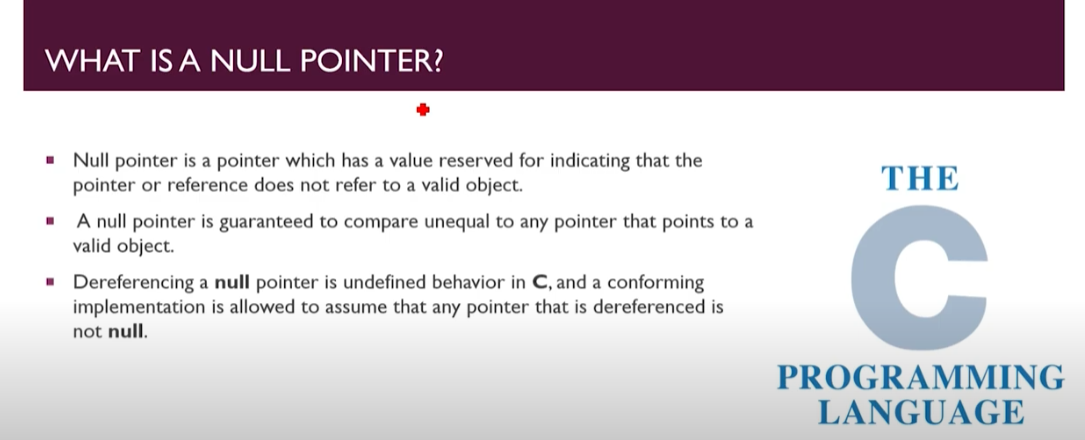
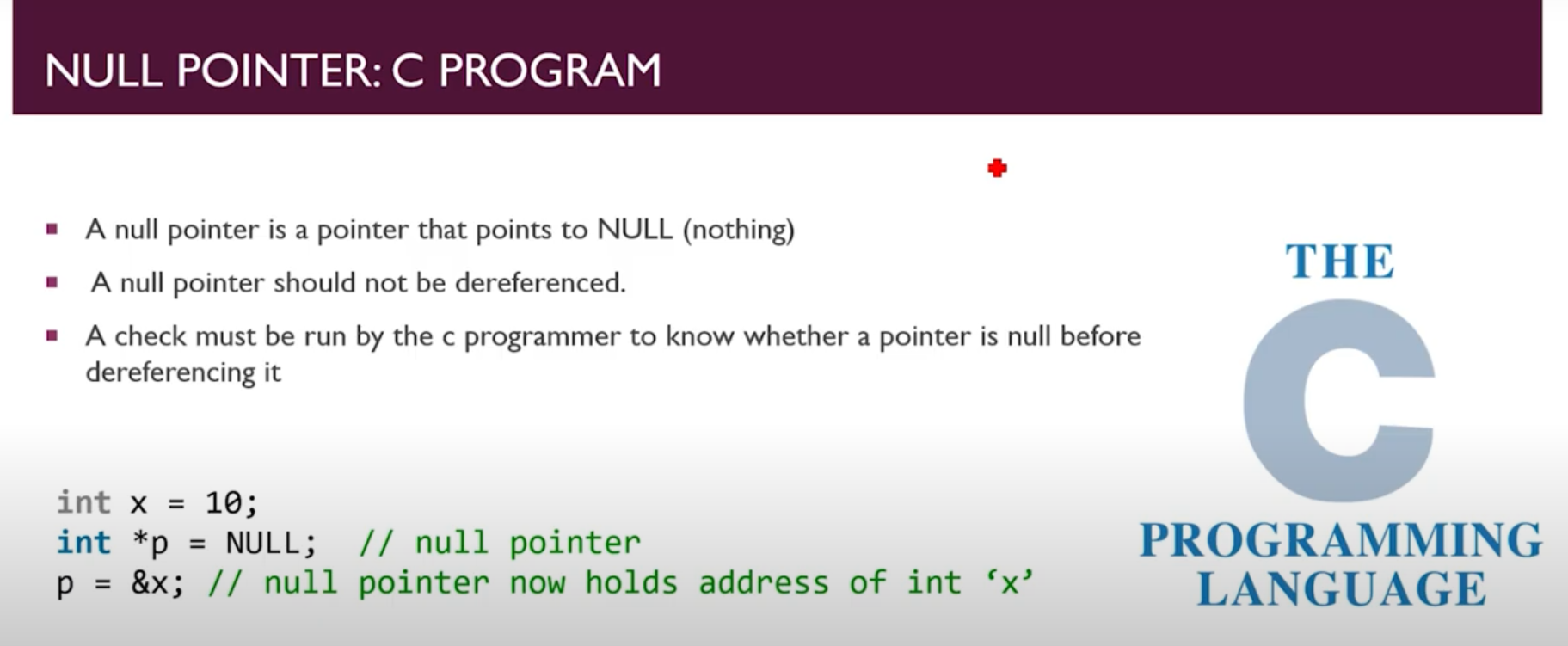
Content 38

NULL Pointer In C Language

The concept of the NULL pointer is very easy and simple to understand. A NULL pointer is a pointer that does not point to any memory location. It generally points to NULL or 0th memory location, so in simple words, no memory is allocated to a NULL pointer.





#### Dereference:

Here again, we will see the concept of dereferencing as its behavior, in this case, is the same as a void pointer. We can say that the type of a NULL pointer is void. So we have to typecast it into any variable the same as in the void pointer case.

#### NULL pointer vs. Uninitialized pointer:

The two are different as the Null pointer points to the 0th memory location, which means that it does not occupy any memory location. In contrast, an uninitialized pointer means that the pointer occupies a garbage value. The garbage value can be any value the garbage collector assigns to the pointer, which may point to some memory location. So to be on the safe side, NULL pointers are preferred.

#### NULL pointer vs. Void pointer:

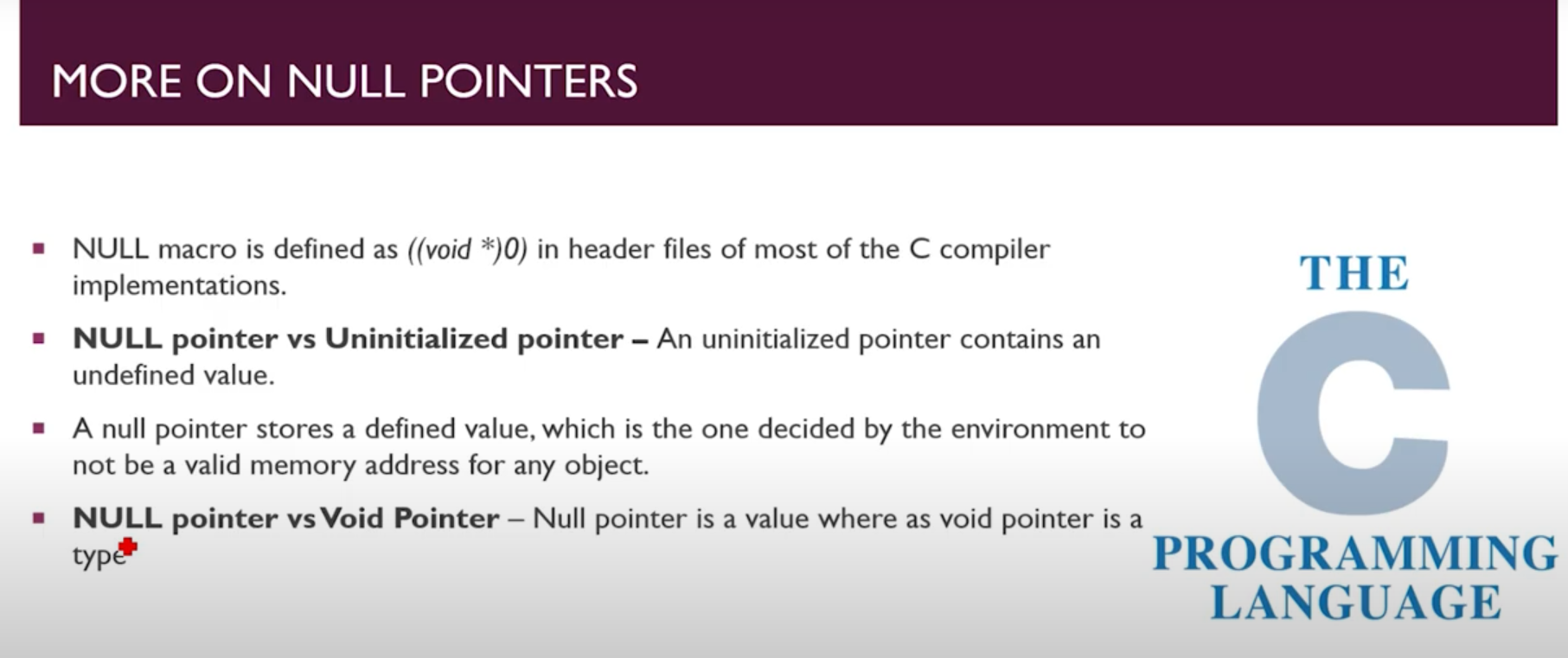
NULL pointer and void pointer may sound similar to their wordy meanings overlap too much, but they are very different as the NULL pointer is a value, and the void pointer is a type. We will see the meaning of the sentence with the help of an example.

int \*ptr = NULL;

Here we have set ptr to NULL, which means it is not pointing to any memory location. Now let us see another example:

void \*ptr;

Now, this is a void pointer in which the value will set according to what sort of value we store in it. If we equal it to an integer, then its value will be int, and if we pass a character variable in it then, its value will be char and so on.



#### Advantages:

* We can initialize the pointer variable without allocating any specific address to it.
* We can use it to check whether a pointer is legitimate or not. We can check that by making the pointer a NULL pointer, after which it can not be dereferenced.
* It is used for comparison with other pointers to check whether they are pointing to some memory address or not.
* We use it for error handling in the case of C programming.
* We can pass a NULL pointer at places where we do not want to pass a pointer with a valid memory address.

**Code1:**

// Understanding Null pointer

#include <stdio.h>

int main(int argc, char const \*argv[])

{

    int a = 34;

    int \*ptr = &a;

    printf("The address of a is %d\n", ptr);

    printf("The value of a is %d", \*ptr);

    return 0;

}

**Output:**

The address of a is 6422216

The value of a is 34

**If I use Null pointer here Then;**

// Understanding Null pointer

#include <stdio.h>

int main(int argc, char const \*argv[])

{

    int a = 34;

    int \*ptr = NULL;

    printf("The address of a is %d\n", ptr);

    printf("The value of a is %d", \*ptr);

    return 0;

}

**Output:**

The address of a is 0

Here I got address as 0 because in most of compilers the value of null pointer is (void \*)0.

But don’t get value because compiler get confused about what to do? And got crash.

**For prevent from Crash we will gonna write above code like:**

**Code for Handling the Crash Error:**

#include <stdio.h>

int main(int argc, char const \*argv[])

{

    int a = 34;

    int \*ptr = NULL;

    if (ptr != NULL)

    {

        printf("The address of a is %d\n", ptr);

        printf("The value of a is %d", \*ptr);

    }

    else{

        printf("The Null pointer is there and it can't be Dereferenced");

    }

    return 0;

}

**Output:**

The Null pointer is there and it can't be Dereferenced

“And If null is replaced by &a then simply ‘if statement’ will execute

and we got a desired output.”