Content 39

Dangling Pointer In C

Dangling:- It means Hanging.

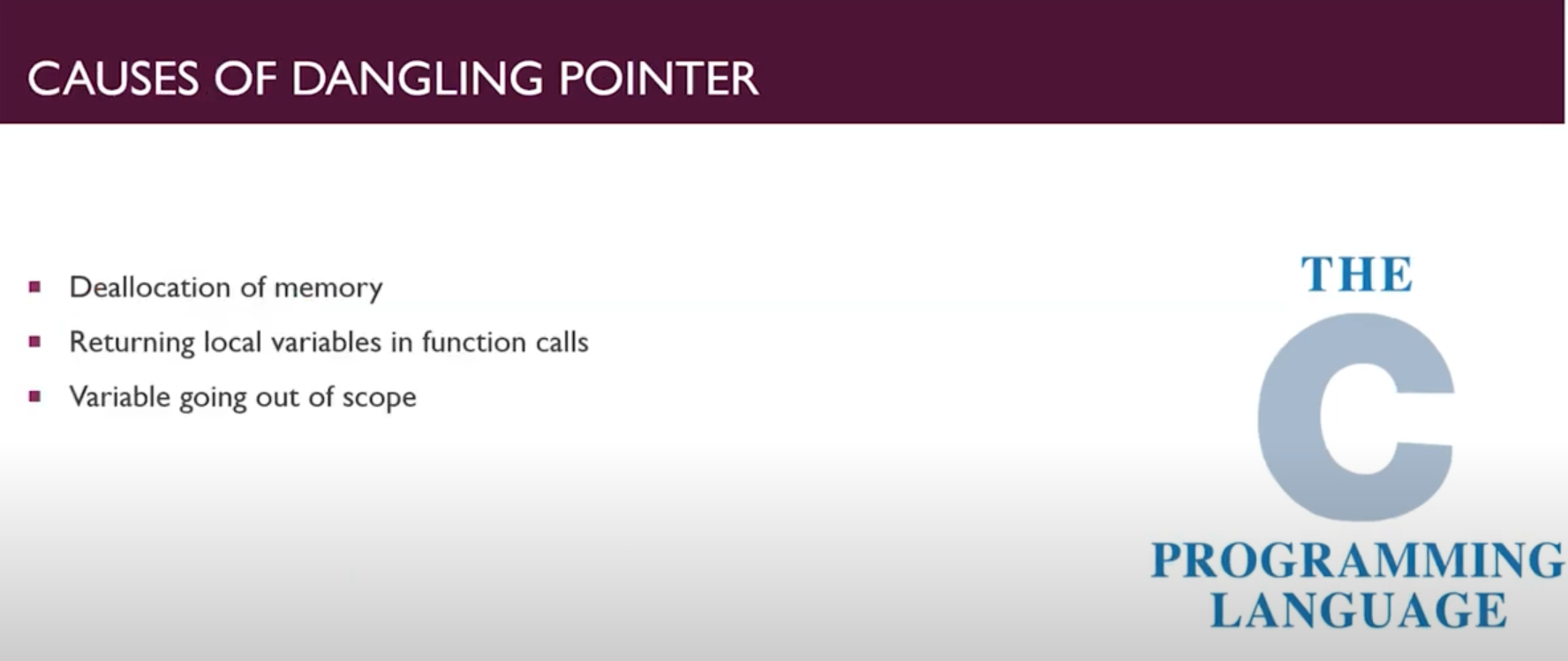
#### C:\Users\Aamaan Satvilkar\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot (30).png

#### What is a dangling pointer?

Dangling pointers are pointers that are pointing to a memory location that has been freed or deleted.

Dangling pointers arise during object destruction, when an object with an incoming reference is deleted or de allocated, without modifying the value of the pointer, so that the pointer still points to the memory location of the de allocated memory.The system may reallocate the previously deleted memory; the unpredicted result may occur as the memory may now contain different data.

“There are some causes of dangling pointers. The causes are explained below with examples”.



#### De-allocating or free variable memory:-

When memory is de allocated, the pointer keeps pointing to freed space.

// De allocation of memory block

#include <stdio.h>

#include <stdlib.h>

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

{

    int \*ptr = (int \*)malloc(5 \* sizeof(int));

    ptr[0]=41;

    ptr[1]=76;

    ptr[2]=31;

    ptr[3]=11;

    free(ptr); // ptr is now become dangling pointer

    return 0;

}

So here we had created pointer with the help of malloc but as malloc returns void so we had used int\* to convert it into int\* pointer.

#### Function call

Now, we will see how the pointer becomes dangling with the function call.

//  Dialing of pointer by returning local Variables by callas

#include <stdio.h>

#include <stdlib.h>

int myfunc(int sum)

{

    int a = 2, b = 6;

    sum = a + b;

    printf("The sum is: %d", sum);

    return sum;

}

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

{

    int \*ptr = myfunc(ptr); //as the the func returns the value, ptr becomes Dangling pointer

    return 0;

}

Here as Function returns the value it becomes the ptr becomes Dangling pointer.

* **Variable goes Out of Scope:**

// Case 3: If a variable goes out of scope

    {

        int a = 12;

        danglingPtr3 = &a;

    }

    // Here variable a goes out of scope which means danglingPtr3 is

    // pointing to a location which is freed and hence danglingPtr3 is now a dangling pointer

#### How to Avoid the Dangling Pointer Errors?

The dangling pointer introduces nasty bugs in our C programming and these bugs frequently become security holes at a time. These dandling pointer errors can be avoided by initializing the pointer value to the NULL. If we assign the **NULL value** to the pointer, then the pointer will not point to the memory location that has been freed. By assigning t he NULL value to the pointer means that the pointer is not pointing to any memory location.

