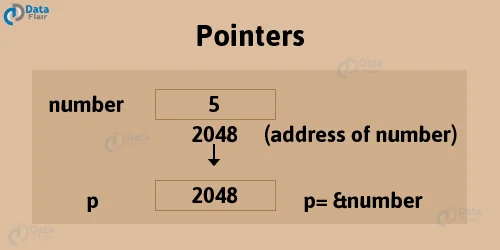
**Pointer**

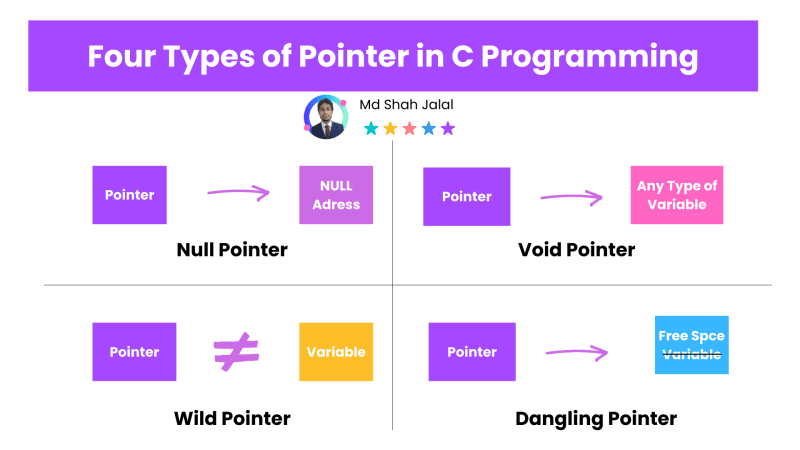
A pointer is a variable whose value is the address of another variable of the same type.



**\* Operator**

The value of the variable that the pointer points to by dereferencing using the \* operator.

**Types of Pointers**



**Null Pointer:**

A pointer that is not assigned any value but NULL is known as the NULL pointer.

If we don't have any address to be specified in the pointer at the time of declaration, we can assign NULL value. It will give a better approach.

Syntax:

Int \*p = NULL;

#include<iostream>

Using namespace std;

int main(){

int \*p = NULL;

cout<<\*p;

}

**Void Pointer:**

When a pointer is declared with a void keyword, then it is called a void pointer. To print the value of this pointer, you need to typecast it.

Syntax:

void \*p;

#include<iostream>

Using namespace std;

int main(){

{

int a=2;

void \*p;

p= &a;

cout<<"After Typecasting, a =", \*(int \*)p);

return 0;

}

**Wild Pointer:**

A wild pointer is only declared but not assigned an address of any variable. They are very tricky, and they'll cause segmentation errors.

#include<iostream>

Using namespace std;

int main(){

int \*p ;

cout<<\*p;

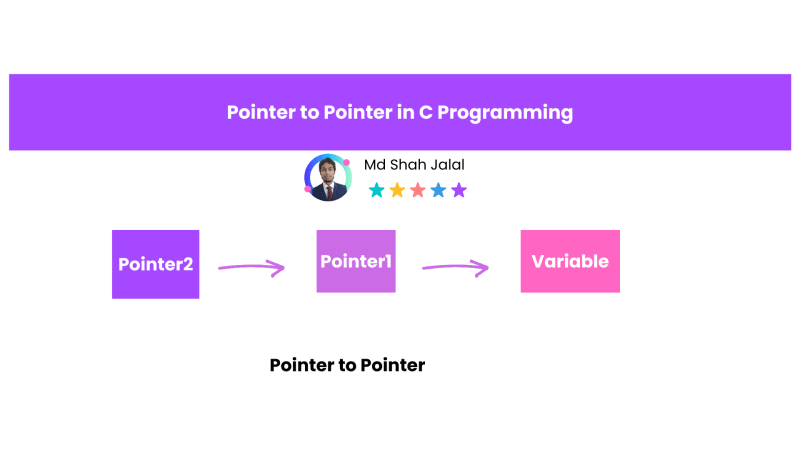
}

**Dangling Pointer:**

A pointer pointing to a memory location that has been deleted (or freed) is called a dangling pointer.

**Pointer to Pointer/Double pointer**

A pointer will indirectly point to a variable via another pointer.



#include<iostream>

Using namespace std;

int main(){

int a = 5;

int \*p = &a;

int \*\*q = &p;

cout<<a<<endl;

cout<<p<<endl;

cout<<q<<endl;

\*\*q = 7;

cout<<a;

return 0;

}