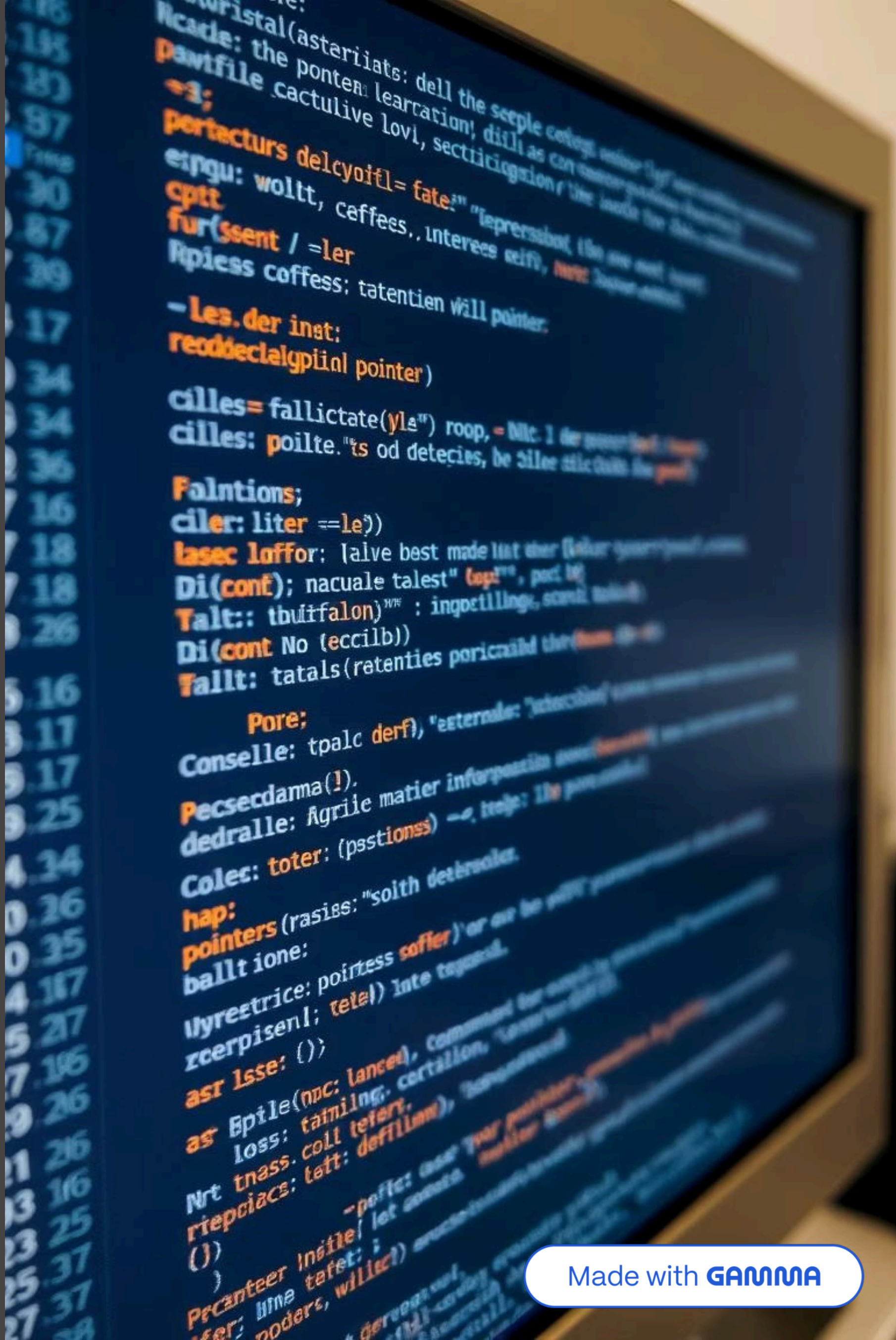
Pointers in C: A Comprehensive Summary

Pointers are essential for memory manipulation in C. They enable dynamic memory allocation and complex data structures. Mastering pointers is critical for efficient C programming.

by Hamdi Emad



What Are Pointers?

Definition

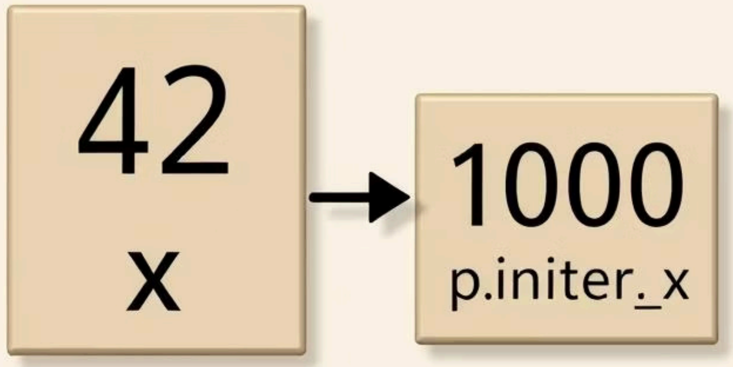
A pointer stores the address of another variable.

Declaration

Use `*` operator, e.g. *int *ptr;*

Address Access

Use `&` operator to get variable address, e.g. ptr = &variable;



Double Pointers (Pointers to Pointers)

Purpose

Stores the address of another pointer.

Declaration

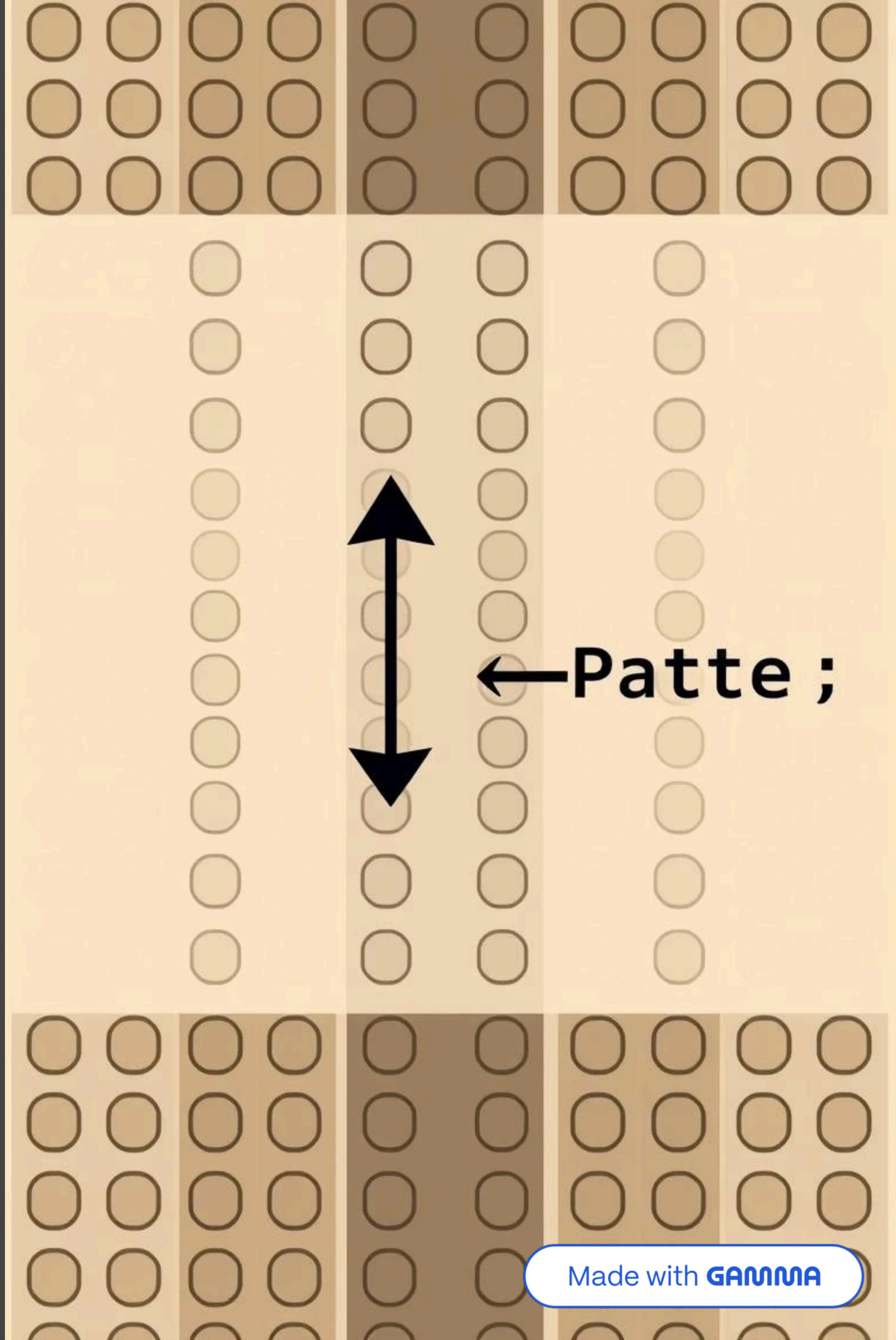
Use double asterisk, e.g. *int **ptr;*

Usage

Modify pointers themselves, not just their data.

Applications

Dynamic arrays of pointers, complex structures.



Purpose of Double Pointers

Modify Pointers in Functions

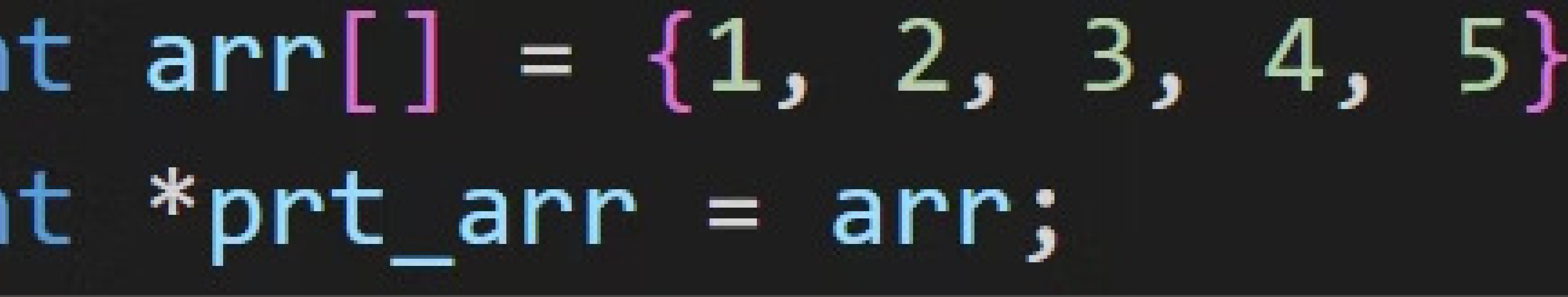
Allows functions to alter pointer values directly.

Dynamic Arrays

Resize arrays like arrays of strings dynamically.

Multi-dimensional Arrays

Dynamically represent arrays like char **array_of_strings;



```
t arr[] = {1, 2, 3, 4, 5}
t *prt_arr = arr;
```

Pointers and Arrays

Array Name

Decays to pointer to first element.

Indexing

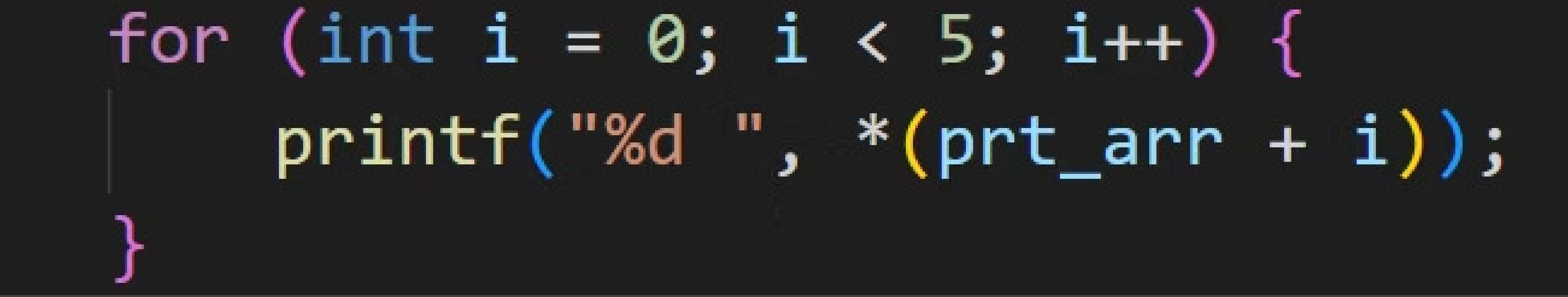
array[i] equals **(array + i)*.

Pointer Use

Efficiently traverse arrays using pointers.

Difference

Pointer is variable; array name is constant address.



```
for (int i = 0; i < 5; i++) {
    printf("%d ", *(prt_arr + i));
}
```

Example: Array Traversal with Pointers

1 **Initialize Array**

int arr = {10, 20, 30, 40, 50};

2 **Pointer Init**

*int *ptr = arr;*

3 **Access Element**

**(ptr + 2)* outputs 30.

4 **Increment Pointer**

Advances to next array element.

Pointers and Strings

Strings as Arrays

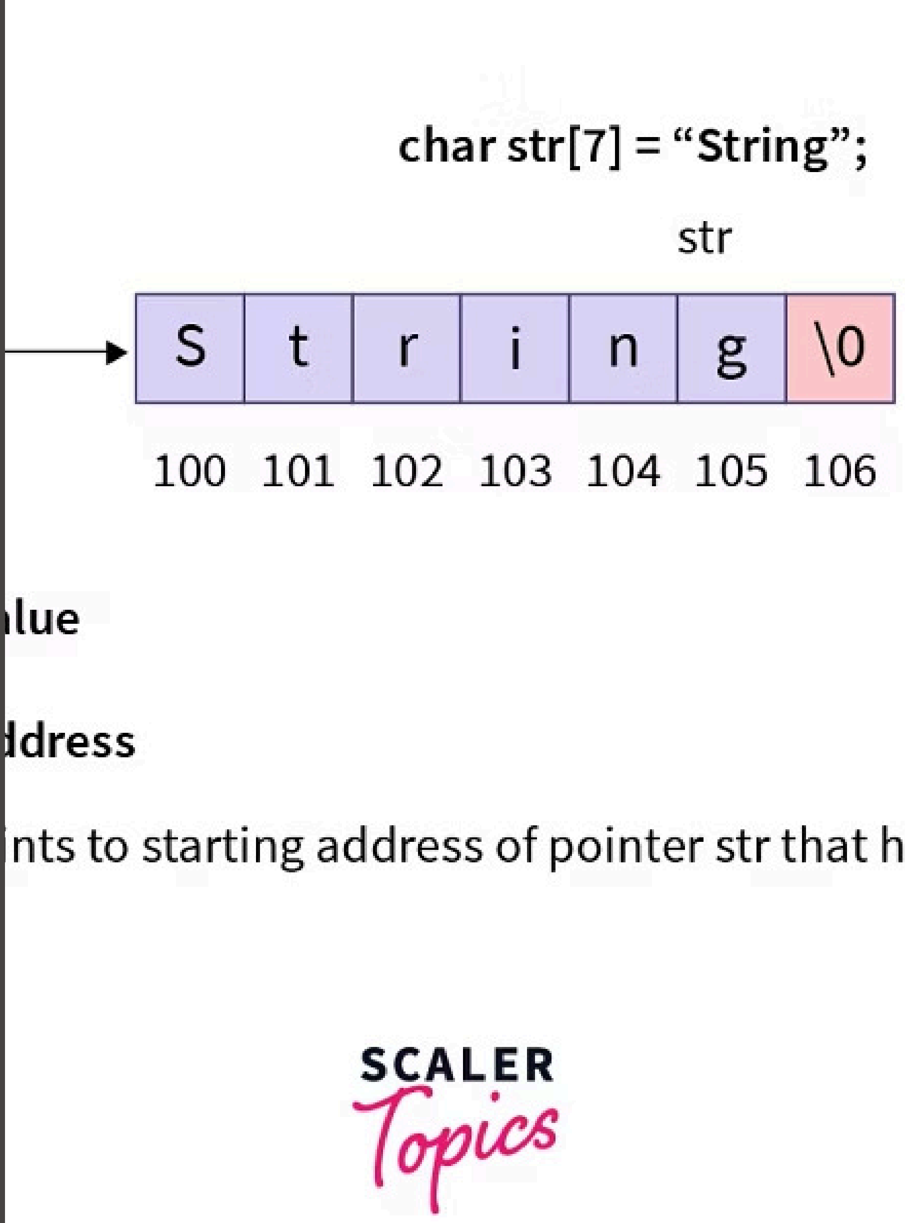
C strings are character arrays terminated by '\ 0'.

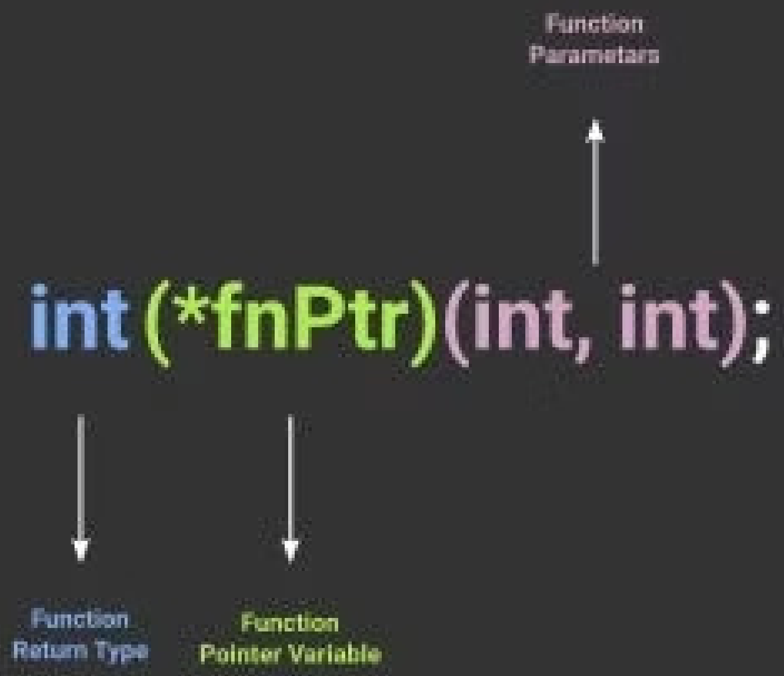
Pointer Usage

Pointers efficiently manipulate strings.

Common Functions

Functions like *strcpy*, *strlen* use pointers.





Function Pointers

Definition

A pointer storing a function's address.

Declaration

Use *int (*func_ptr)(int, int);*

Use

Pass functions as arguments and enable callbacks.

Capabilities

Support dynamic function calls.

Purpose of Function Pointers

Callback Functions

Enable event-driven programming in GUIs.

Generic Algorithms

Work with multiple functions using one codebase.

Dynamic Dispatch

Choose functions at runtime, like *qsort*.

Conclusion

Fundamental Concept

Pointers are core to C language power.

Double Pointers

Enables advanced memory and array management.

Pointers & Arrays

Understand their relationship for efficient code.

Function Pointers

Facilitate flexibility and callbacks.

Best Practices

Use pointers carefully to avoid errors.