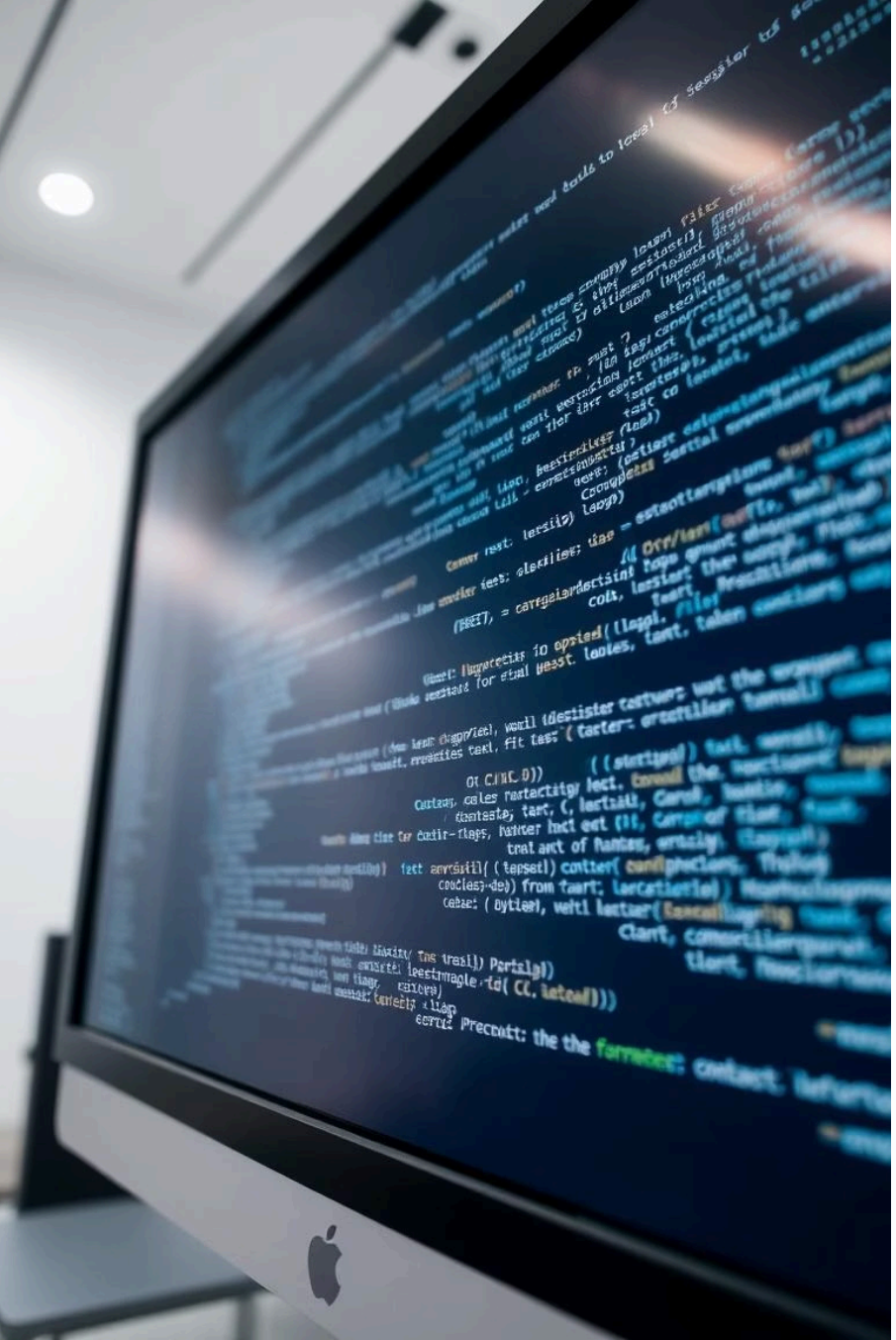


Arrays and Strings in C

Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.

To create an array, define the data type (like `int`) and specify the name of the array followed by **square brackets []**.

To insert values to it, use a comma-separated list inside curly braces, and make sure all values are of the same data type.

by Hamdi Emad

Understanding 1D Arrays

1D arrays store same-type elements contiguously. Declare them with *int arr[5];*. Initialize as *int arr[5] = {1, 2, 3, 4, 5};*. Access elements via *arr[index]*. Indexes start at 0. Let's calculate the average of array elements.

1 **Contiguous Elements**

Same-type elements are stored together.

2 **Declaration**

Specify size and data type.

3 **Zero-Based Indexing**

Array access starts from 0.

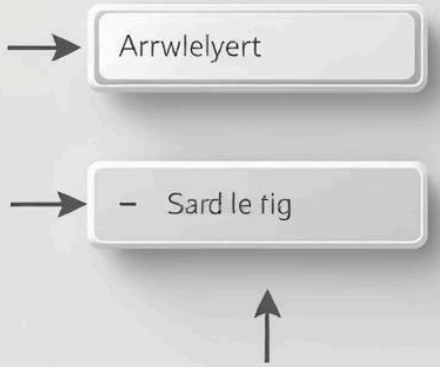
Exploring 2D Arrays

2D arrays are arrays of arrays. They represent tables or matrices. Declare them with *int matrix[3][3];*. Initialize row-wise: *int matrix[2][2] = {{1, 2}, {3, 4}};*. Access elements with *matrix[row][col]*. Let's perform matrix addition.

1 Table Representation

2 Row and Column Index

3 Matrix Operations



1D Array Operations

Traverse arrays using *for* loops. Insert elements carefully considering bounds. Delete elements by shifting. Search using linear or binary search. Sort using bubble or selection sort. These operations manipulate array content.

Searching

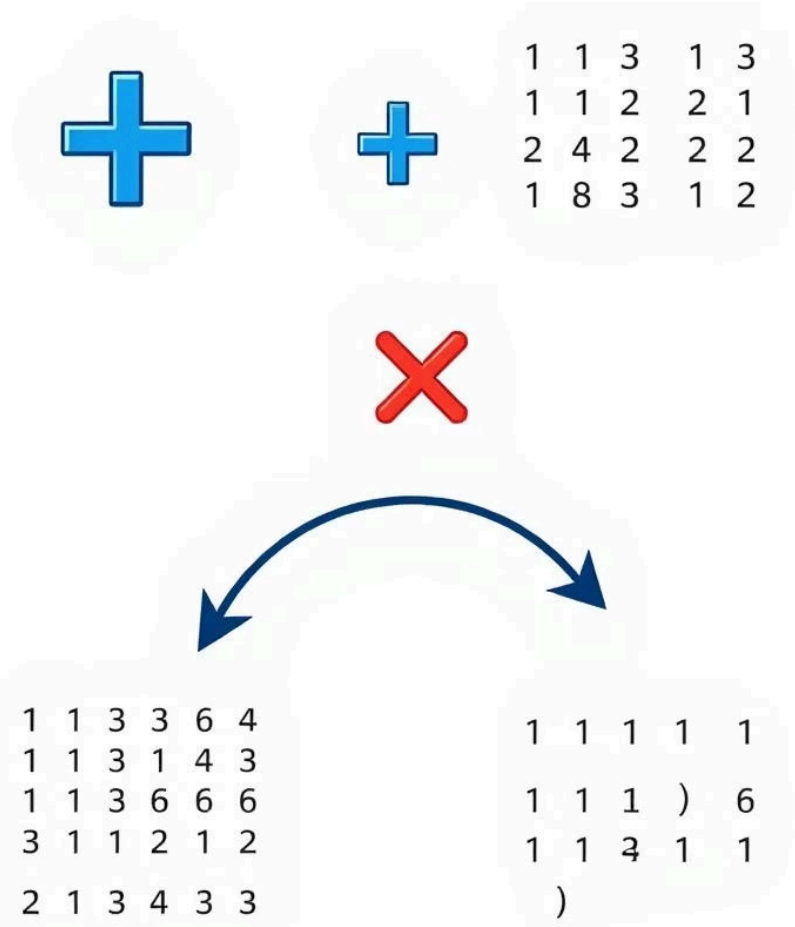
Sorting

Insertion

2D Array Operations

Use nested loops to traverse 2D arrays. Perform matrix addition, subtraction, and multiplication. Transpose matrices by swapping rows and columns. Sum elements in each row or column.

1. Traverse
2. Operate
3. Transpose



Understanding Strings in C

Strings are char arrays, null-terminated (\0). Declare with *char str[10];*. Initialize: *char str[] = "Hello";*. Use *scanf*, *printf*, *gets*, and *puts* for I/O. Example: storing names.

Null Termination

Essential for string recognition.

Character Arrays

Strings are arrays of characters.

Input/Output

Various functions for string handling.

sirrcnonatios

1. string length calculation
2. strintlg
3. stinr →

3. sing copatieration

4. string comparise

String Operations

Calculate length with *strlen()*. Copy with *strcpy()*. Concatenate with *strcat()*. Compare with *strcmp()*. Search for substrings with *strstr()*. You can also convert case.

1. Length
2. Copy
3. Concatenate

Summary and Best Practices

We covered 1D, 2D arrays, strings, and operations. Always check array bounds. Manage memory to avoid leaks. Choose appropriate data structures. Use libraries when possible. Adhering to these practices improves your coding!

1 **Bounds Checking**

Prevent buffer overflows.

2 **Memory Management**

Avoid memory leaks.

3 **Data Structure Selection**

Choose wisely.