CPP NOTES – DAY 12

2-D Arrays

an array of arrays, which can be visualized as a table or grid with rows and columns. Each element in a 2D array is accessed using two indices: one for the row and one for the column.

Condition for Multiplying Two 2D Arrays (Matrices)

If:

- Matrix **A** is of size m × n
- Matrix **B** is of size n × p

Then:

- **A × B** is **defined,** and the result is a matrix **C of size m × p**
- The **number of columns of A (n) must equal the number of rows of B (n)**

Eg:

A = [1 2 3]

[4 5 6]

B = [7 8]

[9 10]

[11 12]

Formula to multiply: *C[i][j] = A[i][0]*B[0][j] + A[i][1]*B[1][j] + A[i][2]*B[2][j]*

ie,

*C[0][0] = 1*7 + 2*9 + 3*11 = 7 + 18 + 33 = 58*

*C[0][1] = 1*8 + 2*10 + 3*12 = 8 + 20 + 36 = 64*

*C[1][0] = 4*7 + 5*9 + 6*11= 28 + 45 + 66 = 139*

*C[1][1] = = 4*8 + 5*10 + 6*12 = 32 + 50 + 72 = 154*

C = [58 64]

[139 154]

To find the size of 2d arr:

```
int rows = sizeof(arr) / sizeof(arr[0]);    // 2 rows
int cols = sizeof(arr[0]) / sizeof(arr[0][0]);  // 3 columns
```

Strings

are sequences/collection of characters used to store text and data.

“Hello” => 5 chars + 1 null char (string should always end up with null char)

Char strArr[20] ; => 1 row with 20 cols. Each char in array of chars(strings) is representing cols.

Initialization: string name = “Gautham” (CPP) / char[] = “Hello” (C style)

Common Function(C-style):

Function	Purpose	Example
strlen(s)	Get length of string	strlen("hello") → 5
strcpy(dest, src)	Copy one string to another	strcpy(a, b)
strcat(a, b)	Concatenate strings	strcat("Hi ", "there")
strcmp(a, b)	Compare strings	strcmp("hi", "hi") → 0
strchr(s, c)	Find first occurrence of a char	strchr("cat", 'a') → "at"
strstr(s, sub)	Find a substring in string	strstr("hello world", "world")

Passing array to functions:

void function(int arr[]); // same as:

void function(int arr);*

FYI

- *String class is designed to make handling text easier, safer, and more powerful than using traditional C-style character arrays (char[] or char*).*
- *Segmantation fault -> occurs when a program tries to access memory it doesn't have permission to.*
  *Eg: int *ptr = NULL;*
  **ptr = 10; // CRASH: invalid memory access*

- *when using scanf() in C, you must use the* **&** *(address-of) operator. This is because scanf() needs the address of the variable where it should store the input value.*

- *For strings (character arrays), you do not use **&** with scanf().*
  Because the array name itself is the pointer to its first element.
  Eg: char name[100];
  scanf("%s", name);