PROGRAMMING FUNDAMENTALS

WEEK 10: 2D ARRAYS AND STRINGS IN C

TABLE OF CONTENTS

2D ARRAY

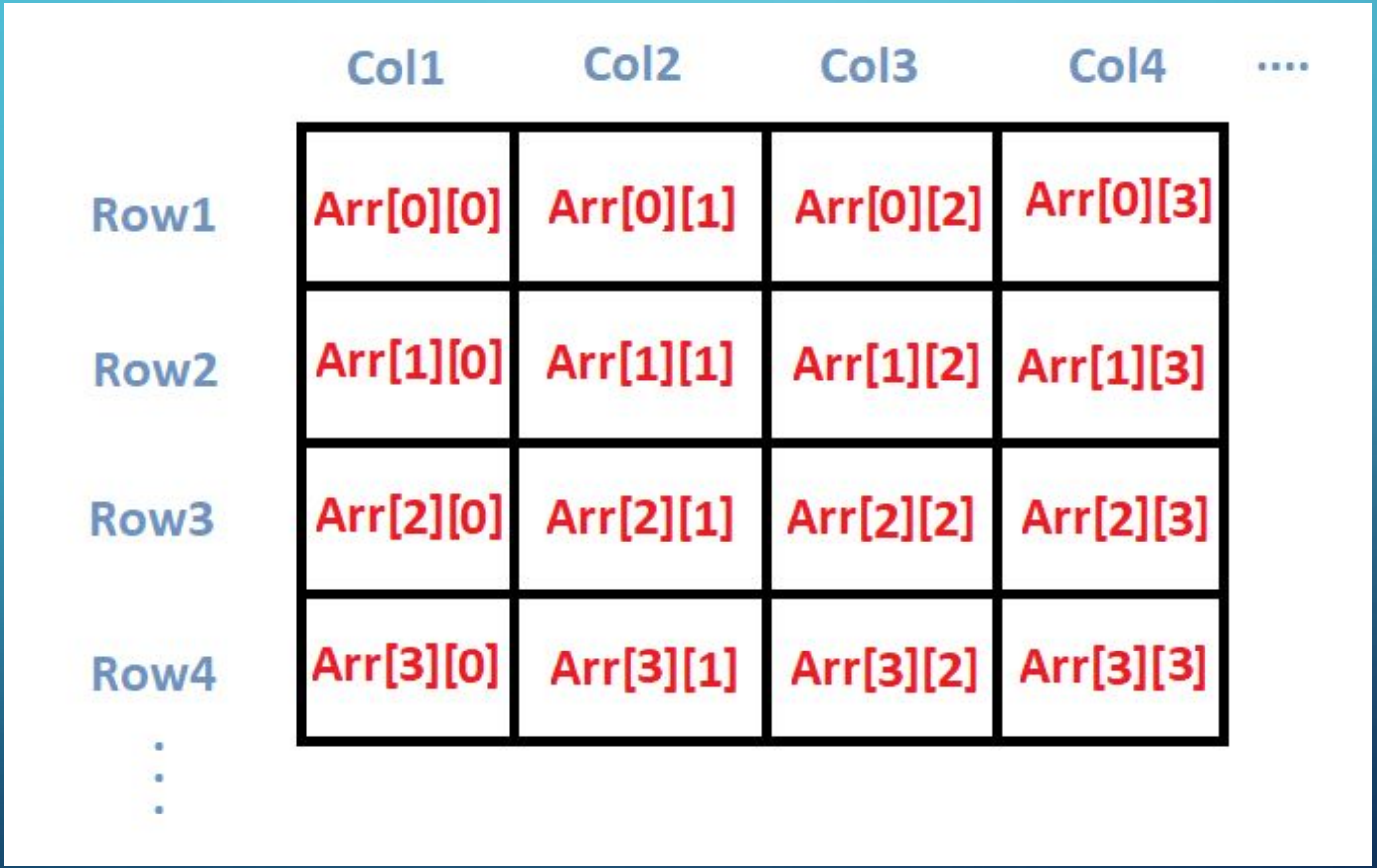
- Array of arrays (rows × columns)

Syntax:

```
int matrix[3][4];  // 3 rows, 4 cols
int mat[2][3] = {{1,2,3}, {4,5,6}};
```

Access: matrix[i][j]

2D ARRAY



	Col1	Col2	Col3	Col4	....
Row1	Arr[0][0]	Arr[0][1]	Arr[0][2]	Arr[0][3]	
Row2	Arr[1][0]	Arr[1][1]	Arr[1][2]	Arr[1][3]	
Row3	Arr[2][0]	Arr[2][1]	Arr[2][2]	Arr[2][3]	
Row4	Arr[3][0]	Arr[3][1]	Arr[3][2]	Arr[3][3]	
⋮					

```
float test_scores[10][4];
```

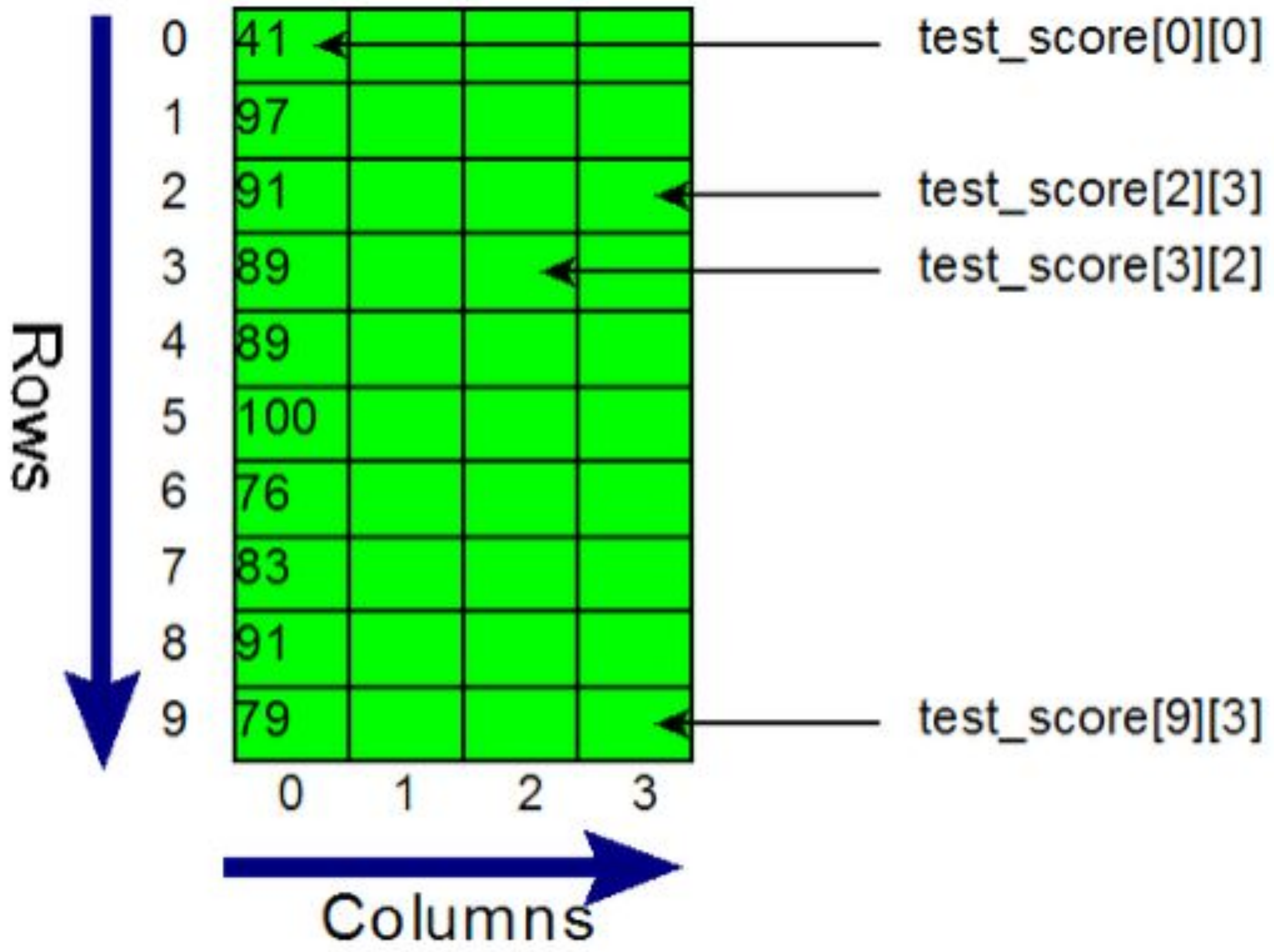


Figure 3. A two-dimensional array. The array definition at the top creates a two-dimensional array of `float` elements or variables arranged as a table with 10 rows and 4 columns. So, the array has space for `10×4 = 40 float` elements. Valid row index values are in the range 0 to 9, and valid column indexes are in the range 0 to 3.

2D ARRAY – PRACTICE QUESTION 1

Problem: Print all elements of a 2D array row-wise.

Input:

```
int mat[2][3] = {{1, 2, 3}, {4, 5, 6}};
```

Output:

```
1 2 3
4 5 6
```

2D ARRAY – PRACTICE QUESTION 1

Nested loops demo:

```
for (int i = 0; i < rows; i++) {
    for (int j = 0; j < cols; j++) {
        printf("%d ", mat[i][j]);
    }
    printf("\n");
}
```

Outer loop = rows, inner loop = columns

2D ARRAY – PRACTICE QUESTION 2

Problem: Find the sum of all elements in a 2D array.

Input:

```
int mat[2][3] = {{1, 2, 3}, {4, 5, 6}};
```

Output: 21

2D ARRAY – PRACTICE QUESTION 2

Tip: Reuse nested loop structure

Tip to compute total elements: rows * cols

STRING FUNCTIONS IN C

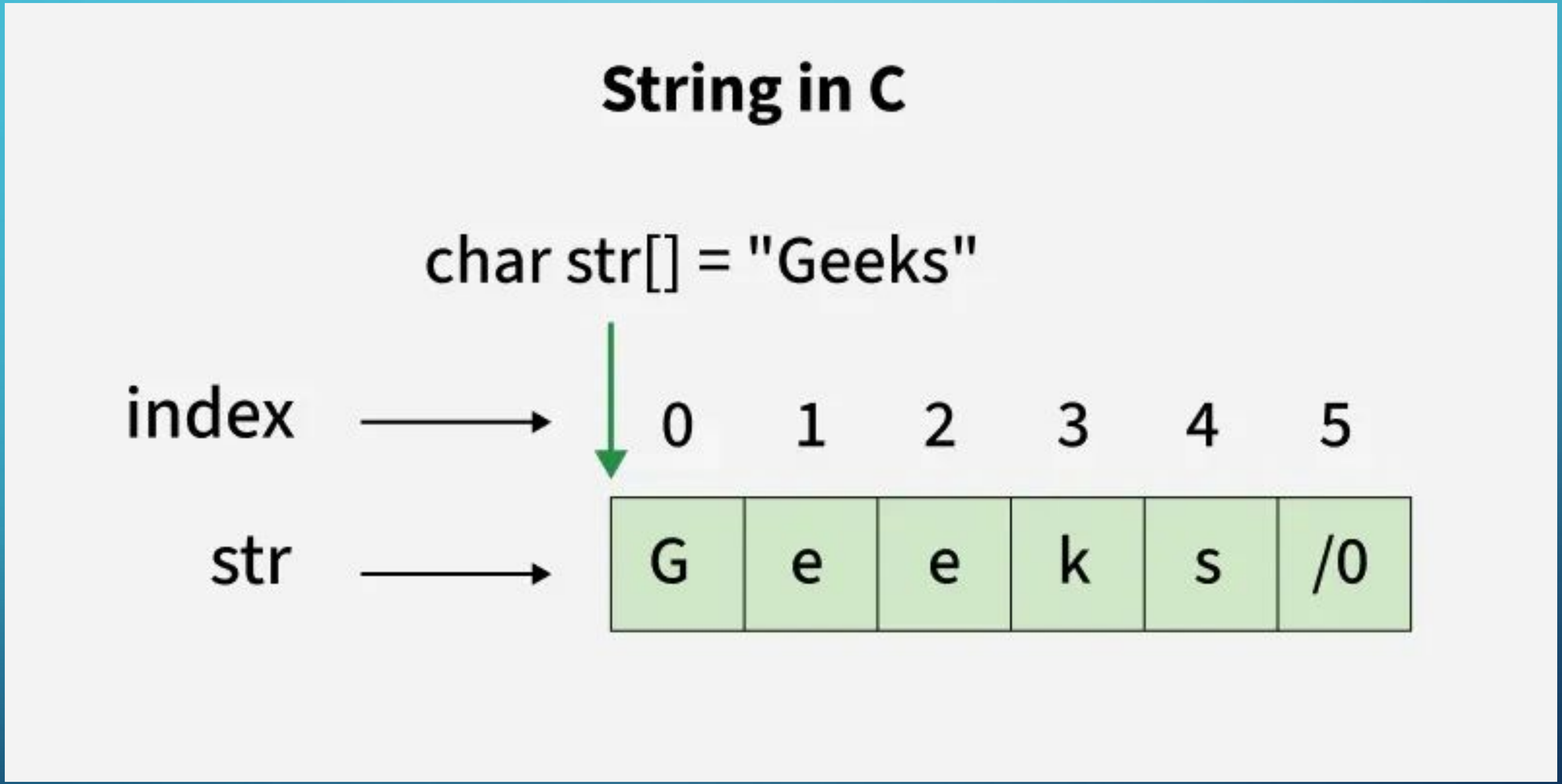
STRCPY, STRCAT, STRCMP, STRLEN, AND MORE

WHAT IS A STRING IN C?

- A string in C is a char array that ends with a null character ‘\0’.
- Example:

```
char name[] = “Ali”;
char name[] = {‘A’, ‘l’, ‘i’, ‘\0’};
```

- Always needs space for ‘\0’.
- Not a built-in type (unlike Python/Java)
- Must include `<string.h>` to use string functions


String in C
char str[] = "Geeks"
index
0
1
2
3
4
5
str
G
e
e
k
s
/0

STRCPY FUNCTION

- Purpose: Copy source string Destination string
- Syntax:

```
strcpy(dest, src);
```

STRCPY FUNCTION – CODE

```c
#include <stdio.h>
#include <string.h>
int main() {
    char src[] = "Hello";
    char dest[10];          // Must be big enough!
    strcpy(dest, src);
    printf("dest = %s\n", dest);  // Output: Hello
    return 0;
}
```

STRNCPY FUNCTION

- Purpose: Copy at most n characters of source string Destination string
- **Safer Copy**
- Syntax:

```
strncpy(dest, src, n);
```

- **Why safer?** Prevents overflow by limiting copy length.

STRNCPY FUNCTION – CODE

```
#include <stdio.h>
#include <string.h>
int main() {
    char src[] = "World";
    char dest[4];  // Only 4 bytes
    strncpy(dest, src, 3);  // Copy 3 chars
    dest[3] = '\0';         // ⚠️ MUST add null terminator!
    printf("%s\n", dest);   // Output: Wor
    return 0;
}
```

STRCAT FUNCTION

- Purpose: append one string to another
- Syntax:

```
strcat(str1, str2);
```

STRCAT FUNCTION – CODE

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[20] = "Hello ";
    char str2[] = "World";
    strcat(str1, str2);  // str1 becomes "Hello World“
    printf(“str1 = %s\n", str);
    return 0;
}
```

STRNCAT FUNCTION

- Purpose: append at most n characters of one string to another.
- Syntax:

```
strncat(str1, str2, n);
```

STRNCAT FUNCTION – CODE

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[10] = "Hi";
    strncat(str1, " there!", 3);  // Appends " th“
    // Result: "Hi th" → automatically adds ‘\0’
    printf(“%s”, str1);
    return 0;
}
```

STRCMP FUNCTION

- Purpose: returns value:
  - 0 → strings are equal
  - < 0 → first string is "less than" second
  - > 0 → first string is "greater than" second
- Syntax:

```
strcmp(str1, str2);
```

STRCMP FUNCTION – CODE

```
#include <stdio.h>
#include <string.h>
int main() {
    if (strcmp("apple", "banana") == 0)
    {
        printf("Same");
    } else {
        printf("Different");  // This prints
    }
    return 0;
}
```

STRNCMP FUNCTION

- Purpose: returns value:
  - 0 → first n characters of strings are equal
  - < 0 → first n characters of first string is "less than" second
  - > 0 → first n characters of first string is "greater than" second
- Syntax:

```
strcmp(str1, str2, n);
```

STRNCMP FUNCTION – CODE

```
#include <stdio.h>
#include <string.h>
int main() {
    if (strncmp("apple", “apply“, 4) == 0)
    {
        printf("Same"); // This prints
    } else {
        printf("Different");
    }
    return 0;
}
```

STRLEN FUNCTION

- Purpose: get string length (returns the number of characters before ‘\0’
- Does not count the null terminator ‘\0’.
- Syntax:

```
strlen(word);
```

STRLEN FUNCTION – CODE

```c
#include <stdio.h>
#include <string.h>
int main() {
    char word[] = "C programming";
    int len = strlen(word);
    printf("Length = %d\n", len);  // Output: 13
    return 0;
}
```

ASSIGNMENT

1. Find sum of diagonal elements in a square 2D array
   Input: {{1,2,3},{4,5,6},{7,8,9}} → Output: 1+5+9 = 15
2. Write a program that copies "Hello" into a new string using strcpy and prints it.
3. Concatenate your first name and last name using strcat.
4. Compare two user-input strings using strcmp and print “Equal” or “Not Equal”.
5. (Challenge) Use strncpy to copy only the first 4 letters of "Programming" into a buffer, and print it safely.

Instructions:
Submit as .c files