📘 Session 17: Strings in C

🎯 Objectives

By the end of this session, you should be able to:

- Understand what strings are in C.
- Use pointers to manipulate and access strings.
- Perform input and output operations with strings.
- Apply standard string functions from `<string.h>`.
- Pass arrays and strings to functions.
- Write simple programs that manipulate strings in C.

📌 1. Concept of Strings in C

Definition:

In C, a string is a sequence of characters terminated by a null character `\0`.

Explanation:

A string is simply a character array. When we define a string like `"Hello"`, it is stored in memory as:

```
'H' 'e' 'l' 'l' 'o' '\0'
```

The null character signifies the end of the string.

Example:

```c
char name[] = "John";
```

This creates a character array with 5 elements: `'J'`, `'o'`, `'h'`, `'n'`, and `\0`.

Classwork:

Declare and initialize a string variable that stores your first name and print it using `printf`.

📌 2. Pointers to Strings

Definition:

A pointer to a string is a pointer to the first character of a null-terminated character array.

Explanation:

Instead of storing the entire string in an array, we can use a pointer to refer to a string literal stored in memory.

Example:

```c
char *name = "Alice";
printf("%s", name);
```

This prints "Alice".

Classwork:

Create a pointer to a string containing your last name and print it.

📌 3. String Input and Output Functions

Definition:

C provides several ways to read and print strings.

Explanation:

Input:

- `scanf("%s", str);` — Reads a word.
- `fgets(str, size, stdin);` — Reads a line safely.

Output:

- `printf("%s", str);` — Prints a string.
- `puts(str);` — Prints a string followed by a newline.

Example:

```
char name[30];
printf("Enter your name: ");
fgets(name, 30, stdin);
puts(name);
```

Classwork:

Write a program that asks the user to enter their favorite food and prints it.

📌 4. Various String Functions

Definition:

The `<string.h>` library provides built-in functions to manipulate strings.

Explanation & Examples:

Function	Description	Example
`strlen(str)`	Returns length of string	`int len = strlen("Hello");`
`strcpy(dest, src)`	Copies one string to another	`strcpy(name, "Bob");`
`strcat(dest, src)`	Concatenates two strings	`strcat(greet, name);`

Function	Description	Example
`strcmp(a, b)`	Compares two strings	`if(strcmp(a,b)==0)`
`strchr(str, ch)`	Finds a character in a string	`strchr(str, 'a');`

Classwork:

Write a program to:

- Read two strings from user
- Compare them using `strcmp`
- Print whether they are equal or not

📌 5. Passing Arrays as Function Arguments

Definition:

Arrays in C are passed to functions as pointers to their first element.

Explanation:

When you pass an array, you're actually passing the address of its first element.

Example:

```
void printArray(int arr[], int size) {
    for(int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
}
```

Classwork:

Write a function that accepts an array of integers and prints all its values.

📌 6. Passing Strings as Function Arguments

Definition:

You can pass a string to a function the same way you pass an array — as a pointer.

Explanation:

Since strings are arrays, functions can receive them as `char str[]` or `char *str`.

Example:

```
void greet(char name[]) {
    printf("Hello, %s\n", name);
}
```

Classwork:

Write a function that takes a string and prints "Welcome, [name]!"

🧪 Lab Practice – TL13

Practice all the concepts from this session:

- String declarations and input
- Using pointers to strings
- Using string functions from `<string.h>`
- Passing strings to functions
- Implementing small string-based programs

✅ Summary

- Strings are character arrays ending with `\0`.
- Pointers can simplify string handling.

- Use `<string.h>` for common string tasks.
- You can pass strings/arrays to functions easily in C.

🧠 XP – Session 17
📚 TG – Session 17
🧪 Lab – C-TL13