Day_7 C-Programming

(RECAP OF PREVIOUS DAY)
Strings: Introduction, initializing strings, accessing string elements, Array of strings, Passing strings to functions, String functions like strlen, strcat, strcmp, strcpy, strrev and other string functions.

1. Introduction to Strings

- A string in C is a one-dimensional array of characters terminated by a null character (`\0`).
- Strings are used to store text data.

Example:

```
char str[] = "Hello, World!";
```

Here:

- `str` is a string of size 14 (13 characters + 1 null character `\0`).

---

2. Initializing Strings

Strings can be initialized in multiple ways:

Method 1: Using a String Literal

```
char str1[] = "Hello";
```

Method 2: Character by Character

```
char str2[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
```

Method 3: Using `scanf` or `gets` (deprecated)

```
char str3[20];
scanf("%s", str3);  // Note: Stops reading at whitespace
```

---

3. Accessing String Elements

- Strings are arrays, so elements can be accessed using indices.

Example:

```
char str[] = "C Programming";
printf("First character: %c\n", str[0]);  // Output: C
printf("Fifth character: %c\n", str[4]);  // Output: r
```

---

4. Array of Strings

An array of strings is a 2D array where each row represents a string.

Example:

```
#include <stdio.h>

int main() {
    char colors[3][10] = {"Red", "Green", "Blue"};
    for (int i = 0; i < 3; i++) {
        printf("Color %d: %s\n", i + 1, colors[i]);
    }
    return 0;
}
```

Output:

```
Color 1: Red
Color 2: Green
Color 3: Blue
```

---

5. Passing Strings to Functions

Strings can be passed to functions as arguments.

Example:

```
#include <stdio.h>
```

```
void printString(char str[]) {
    printf("String: %s\n", str);
}

int main() {
    char message[] = "Hello, Functions!";
    printString(message);
    return 0;
}
```

Output:

```
String: Hello, Functions!
```

---

6. Common String Functions in C

The `<string.h>` library provides various functions for string manipulation.

1. `strlen`: Calculate the length of a string

```
#include <stdio.h>
#include <string.h>

int main() {
    char str[] = "C Programming";
    printf("Length: %zu\n", strlen(str));  // Output: 13
    return 0;
}
```

2. `strcpy`: Copy one string to another

```
#include <stdio.h>
#include <string.h>

int main() {
    char source[] = "Hello";
    char destination[10];
    strcpy(destination, source);
    printf("Copied String: %s\n", destination);  // Output: Hello
    return 0;
}
```

3. `strcat`: Concatenate two strings

```
#include <stdio.h>
#include <string.h>

int main() {
    char str1[20] = "Hello";
    char str2[] = " World";
    strcat(str1, str2);
    printf("Concatenated String: %s\n", str1);  // Output: Hello World
    return 0;
}
```

4. `strcmp`: Compare two strings

```
#include <stdio.h>
#include <string.h>

int main() {
    char str1[] = "Apple";
    char str2[] = "Orange";
    int result = strcmp(str1, str2);
    if (result == 0)
        printf("Strings are equal\n");
    else if (result < 0)
        printf("%s comes before %s\n", str1, str2);
    else
        printf("%s comes after %s\n", str1, str2);
    return 0;
}
```

5. `strrev`: Reverse a string

`strrev` is not part of the standard C library but is available in some compilers (e.g., Turbo C). You can implement it yourself:

```
#include <stdio.h>
#include <string.h>

void reverseString(char str[]) {
    int n = strlen(str);
    for (int i = 0; i < n / 2; i++) {
        char temp = str[i];
        str[i] = str[n - i - 1];
        str[n - i - 1] = temp;
    }
}

int main() {
    char str[] = "Reverse Me";
    reverseString(str);
    printf("Reversed String: %s\n", str);  // Output: eM esreveR
    return 0;
}
```

6. `strncpy`: Copy first N characters

```
#include <stdio.h>
#include <string.h>

int main() {
    char source[] = "Hello";
    char destination[10];
    strncpy(destination, source, 3);
    destination[3] = '\0';  // Ensure null termination
    printf("Copied String: %s\n", destination);  // Output: Hel
    return 0;
}
```

7. `strchr`: Find the first occurrence of a character

```
#include <stdio.h>
#include <string.h>

int main() {
    char str[] = "Locate the first 't'.";
    char *pos = strchr(str, 't');
    if (pos)
        printf("Found 't' at position: %ld\n", pos - str);
    else
        printf("Character not found\n");
    return 0;
}
```

8. `strstr`: Find a substring

```
#include <stdio.h>
#include <string.h>

int main() {
    char str[] = "Find the substring here.";
    char *substr = strstr(str, "substring");
    if (substr)
        printf("Substring found at position: %ld\n", substr - str);
    else
        printf("Substring not found\n");
    return 0;
}
```

---

7. Summary of String Functions

Function	Purpose

`strlen`	Calculate the length of a string
`strcpy`	Copy one string to another
`strncpy`	Copy the first N characters of a string
`strcat`	Concatenate two strings
`strcmp`	Compare two strings
`strchr`	Find the first occurrence of a character
`strstr`	Find a substring within a string
`strrev`	Reverse a string (if supported)

Program to input a string and print it:

```
#include <stdio.h>
int main() {
    char str[100];
    printf("Enter a string: ");
    scanf("%s", str);
    printf("You entered: %s\n", str);
    return 0;
}
```

Program to count the number of characters in a string (without `strlen`):

```
#include <stdio.h>
int main() {
    char str[100];
    int count = 0;
    printf("Enter a string: ");
    scanf("%s", str);
    while (str[count] != '\0') {
        count++;
    }
    printf("Length of the string: %d\n", count);
    return 0;
}
```

Program to reverse a string:

```
#include <stdio.h>
#include <string.h>
int main() {
    char str[100], temp;
    int i, j;
    printf("Enter a string: ");
    scanf("%s", str);
    int n = strlen(str);
    for (i = 0, j = n - 1; i < j; i++, j--) {
        temp = str[i];
        str[i] = str[j];
        str[j] = temp;
    }
    printf("Reversed string: %s\n", str);
    return 0;
}
```

Program to check if a string is a palindrome:

```
#include <string.h>
int main() {
    char str[100];
```

```c
    int i, n, flag = 1;
    printf("Enter a string: ");
    scanf("%s", str);
    n = strlen(str);
    for (i = 0; i < n / 2; i++) {
        if (str[i] != str[n - i - 1]) {
            flag = 0;
            break;
        }
    }
    if (flag)
        printf("The string is a palindrome.\n");
    else
        printf("The string is not a palindrome.\n");
    return 0;
}
```

String Functions

Program to implement `strlen`:

```c
#include <stdio.h>
int stringLength(char str[]) {
    int count = 0;
    while (str[count] != '\0') {
        count++;
    }
    return count;
}
int main() {
    char str[100];
    printf("Enter a string: ");
    scanf("%s", str);
    printf("Length of the string: %d\n", stringLength(str));
    return 0;
}
```

Program to concatenate two strings (without `strcat`):

```
#include <stdio.h>
void stringConcat(char str1[], char str2[]) {
    int i = 0, j = 0;
    while (str1[i] != '\0') {
        i++;
    }
    while (str2[j] != '\0') {
        str1[i] = str2[j];
        i++;
        j++;
    }
    str1[i] = '\0';
}
int main() {
    char str1[100], str2[50];
    printf("Enter first string: ");
    scanf("%s", str1);
    printf("Enter second string: ");
    scanf("%s", str2);
    stringConcat(str1, str2);
    printf("Concatenated string: %s\n", str1);
    return 0;
}
```

Program to compare two strings (without `strcmp`):

```c
#include <stdio.h>
int stringCompare(char str1[], char str2[]) {
    int i = 0;
    while (str1[i] != '\0' && str2[i] != '\0') {
        if (str1[i] != str2[i]) {
            return str1[i] - str2[i];
        }
        i++;
    }
    return str1[i] - str2[i];
}
int main() {
    char str1[100], str2[100];
    printf("Enter first string: ");
    scanf("%s", str1);
    printf("Enter second string: ");
    scanf("%s", str2);
    int result = stringCompare(str1, str2);
    if (result == 0)
        printf("Strings are equal.\n");
    else if (result < 0)
        printf("String 1 comes before String 2.\n");
    else
        printf("String 1 comes after String 2.\n");
    return 0;
}
```

Program to find the first occurrence of a character (using `strchr`):

```c
#include <stdio.h>
#include <string.h>
int main() {
    char str[100], ch;
    printf("Enter a string: ");
    scanf("%s", str);
    printf("Enter the character to search for: ");
    scanf(" %c", &ch);
    char *pos = strchr(str, ch);
    if (pos)
        printf("Character found at position: %ld\n", pos - str);
    else
        printf("Character not found.\n");
    return 0;
}
```

Program to find all occurrences of a substring (using `strstr`):

```
#include <stdio.h>
#include <string.h>
int main() {
    char str[100], substr[50];
    printf("Enter the main string: ");
    scanf("%s", str);
    printf("Enter the substring to search: ");
    scanf("%s", substr);
    char *pos = strstr(str, substr);
    if (!pos) {
        printf("Substring not found.\n");
        return 0;
    }
    while (pos) {
        printf("Substring found at position: %ld\n", pos - str);
        pos = strstr(pos + 1, substr);
    }
    return 0;
}
```

Program to count vowels, consonants, digits, and spaces in a string:

```
#include <stdio.h>
int main() {
    char str[100];
    int vowels = 0, consonants = 0, digits = 0, spaces = 0;
    printf("Enter a string: ");
    scanf(" %[^\n]", str);
    for (int i = 0; str[i] != '\0'; i++) {
        if ((str[i] >= 'a' && str[i] <= 'z') || (str[i] >= 'A' &&
str[i] <= 'Z')) {
            if (str[i] == 'a' || str[i] == 'e' || str[i] == 'i' ||
str[i] == 'o' || str[i] == 'u' ||
                str[i] == 'A' || str[i] == 'E' || str[i] == 'I' ||
str[i] == 'O' || str[i] == 'U') {
                vowels++;
            } else {
                consonants++;
            }
        } else if (str[i] >= '0' && str[i] <= '9') {
            digits++;
        } else if (str[i] == ' ') {
            spaces++;
        }
    }
    printf("Vowels: %d, Consonants: %d, Digits: %d, Spaces: %d\n",
vowels, consonants, digits, spaces);
    return 0;
}
```

Practice programs (Home Work)

1. Write a program to find the frequency of each character in a given string.
2. Write a program to count the number of words in a string.
3. Write a program to remove all vowels from a string.
4. Write a program to check if two strings are anagrams of each other.

5. Write a program to find the second most frequent character in a string.
6. Write a program to replace all occurrences of a given character with another character in a string.
7. Write a program to find and replace a substring within a string.
8. Write a program to reverse each word in a string while keeping the word order intact.
9. Write a program to find the longest word in a string.
10. Write a program to convert all lowercase letters in a string to uppercase and vice versa.
11. Write a program to count the number of palindromic substrings in a string.
12. Write a program to sort the characters of a string in alphabetical order.
13. Write a program to check if a string contains only unique characters.
14. Write a program to implement your own version of `strtok` to tokenize a string based on a given delimiter.
15. Write a program to calculate the edit distance (Levenshtein distance) between two strings.
16. Write a program to dynamically allocate memory for a string and reverse it.
17. Write a program to merge two strings alternatively (e.g., merge "abc" and "123" to get "a1b2c3").
18. Write a program to remove duplicate characters from a string.
19. Write a program to remove all whitespace characters from a string.
20. Write a program to find the lexicographically smallest and largest substrings of a string.
21. Write a program to implement `strncmp` to compare the first `n` characters of two strings.
22. Write a program to implement `strncat` to concatenate the first `n` characters of one string to another.
23. Write a program to implement `memcpy` for copying a specified number of bytes between two memory locations.
24. Write a program to implement `strrchr` to find the last occurrence of a character in a string.
25. Write a program to implement `strpbrk` to find the first occurrence of any character from one string in another string.
26. Write a program to check if a string follows a given pattern (e.g., "aabb" matches pattern "xxyy").
27. Write a program to count the number of occurrences of a substring in a string.
28. Write a program to find all permutations of a given string.
29. Write a program to implement the KMP (Knuth-Morris-Pratt) string matching algorithm.
30. Write a program to implement the Rabin-Karp string matching algorithm.
31. Write a program to generate all possible substrings of a string.
32. Write a program to compress a string using Run Length Encoding (e.g., "aaabbc" becomes "a3b2c1").
33. Write a program to check if a string can be rearranged to form a palindrome.
34. Write a program to remove all characters from the first string that are present in the second string.
35. Write a program to count the number of times each word appears in a sentence.

36. Write a program to reverse the order of words in a sentence (e.g., "C is fun" becomes "fun is C").
37. Write a program to find the first non-repeating character in a string.
38. Write a program to calculate the longest prefix that is also a suffix in a string.
39. Write a program to check if a string can be segmented into a space-separated sequence of dictionary words.
40. Write a program to generate all possible valid permutations of parentheses for a given n.
41. Write a program to encode a string by shifting characters by a given value (Caesar Cipher).
42. Write a program to decode a Caesar Cipher string.
43. Write a program to validate if a string is a valid email address.
44. Write a program to parse a CSV file and extract specific data fields from it.
45. Write a program to determine if a string is a valid hexadecimal number.