Trainer: Asif Nafees

Strings in C Language

In C language, strings are a sequence of characters ending with a special character '\0' (null character). Unlike other languages, there is no specific data type for strings in C. Instead, strings are represented as arrays of characters.

1. Declaring a String

There are two main ways to declare a string in C:

```
// Using character array
char str1[20] = "Hello";
// Using pointer
char *str2 = "World";
```

In the first example, str1 is an array of characters that can hold up to 20 characters including the null character. In the second example, str2 is a pointer to a string literal, which is read-only.

2. Initializing a String

A string can be initialized in different ways:

```
char str1[] = "Programming";
char str2[] = {'C', 'o', 'd', 'e', '\0'};
```

Both str1 and str2 represent the same string "Code".

3. Reading a String

Reading Strings with Spaces in C Language

In C, reading input using scanf() has a limitation: it stops reading when it encounters a space, tab, or newline character. This means if the user enters a string with spaces, only the part before the first space is captured.

Example using scanf()

```
#include <stdio.h>
int main() {
    char str[50];
    printf("Enter a string: ");
    scanf("%s", str);
    printf("You entered: %s\n", str);
    return 0;
}
```

Input:
Hello World
Output:
You entered: Hello

As you can see, scanf() only reads "Hello" and stops at the space.

Reading a String with Spaces
To read a string that includes spaces, you can use gets() or fgets().
1. Using gets() (Not Recommended)
The gets() function can read an entire line of input, including spaces, but it has been deprecated due to potential security issues (buffer overflow). Avoid using it in modern C programming.

```
#include <stdio.h>
int main() {
    char str[50];
    printf("Enter a string: ");
    gets(str);  // Not safe
    printf("You entered: %s\n", str);
    return 0;
}
```

Input:
Hello World
Output:
You entered: Hello World
Warning: Using gets() can cause buffer overflow, as it does not check the size of the buffer.

2. Using fgets() (Recommended)
The fgets() function is a safer alternative to gets(). It reads a line of input, including spaces, and stops when it encounters a newline or reaches the specified buffer size.

```
#include <stdio.h>
int main() {
    char str[50];
    printf("Enter a string: ");
    fgets(str, sizeof(str), stdin);
    printf("You entered: %s", str);
    return 0;
}
```

Input:
Hello World from C
Output:
You entered: Hello World from C

Explanation:
fgets(str, sizeof(str), stdin) reads up to sizeof(str) - 1 characters from the input, including spaces, and stores it in str.

It also includes the newline character (\n) at the end of the string, which can be removed if needed.

4. Printing a String
To print a string in C, you can use the printf() function with the %s format specifier. The %s tells the compiler that the argument is a string.

Example of Printing a String
```
#include <stdio.h>
int main() {
    char str[] = "Hello, World!";
    printf("The string is: %s\n", str);
    return 0;
}
```

Output:
The string is: Hello, World!

Explanation
%s is used as a placeholder for the string variable.
str is the string to be printed.

Printing a String Using puts()
You can also use the puts() function to print a string. It automatically adds a newline character at the end.

Example Using puts()
```
#include <stdio.h>
int main() {
    char str[] = "C Programming";
    puts(str);
    return 0;
}
```

Output:
C Programming

Differences Between printf() and puts()
Example of Printing Multiple Strings
```
#include <stdio.h>
int main() {
    char str1[] = "Hello";
```

```
    char str2[] = "World";
    printf("%s, %s!\n", str1, str2);
    return 0;
}
```

Output:
Hello, World!
In this example, %s is used twice to print two different strings.

Important Points
Make sure the string is null-terminated (ends with '\0'), as the %s specifier in printf() relies on it.

5. String Functions in C

Predefined String Functions in C
In C language, the <string.h> library provides several predefined functions for performing various string operations like copying, concatenation, comparison, and manipulation. Here is a list of commonly used string functions with examples.

1. strlen() - String Length
Definition: Returns the length of the string (number of characters before the null character '\0').

```
#include <stdio.h>
#include <string.h>
int main() {
    char str[] = "Programming";
    int length = strlen(str);
    printf("Length of the string: %d\n", length);
    return 0;
}
```

Output:
Length of the string: 11

2. strcpy() - String Copy
Definition: Copies the source string (src) to the destination string (dest).

```
#include <stdio.h>
#include <string.h>
int main() {
    char src[] = "Hello";
    char dest[20];

    strcpy(dest, src);
    printf("Copied String: %s\n", dest);
```

```
    return 0;
}
```

Output:
Copied String: Hello

3. strncpy() - Copy N Characters
Definition: Copies a specified number of characters from source to destination.

```
#include <stdio.h>
#include <string.h>
int main() {
    char src[] = "HelloWorld";
    char dest[20];
    strncpy(dest, src, 5);
    dest[5] = '\0';  // Adding null character manually
    printf("Copied String (5 chars): %s\n", dest);
    return 0;
}
```

Output:
Copied String (5 chars): Hello

4. strcat() - String Concatenation
Definition: Appends the source string (src) to the destination string (dest).

```
#include <stdio.h>
#include <string.h>
int main() {
    char dest[30] = "Hello, ";
    char src[] = "World!";
    strcat(dest, src);
    printf("Concatenated String: %s\n", dest);
    return 0;
}
```

Output:
Concatenated String: Hello, World!

5. strncat() - Concatenate N Characters
Definition: Appends a specified number of characters from the source string to the destination string.

```
#include <stdio.h>
#include <string.h>
int main() {
    char dest[30] = "Hello, ";
```

```
    char src[] = "World!";
    strncat(dest, src, 3);
    printf("Concatenated String (3 chars): %s\n", dest);
    return 0;
}
```

Output:
Concatenated String (3 chars): Hello, Wor

6. strcmp() - String Comparison
Definition: Compares two strings lexicographically.

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[] = "Apple";
    char str2[] = "Banana";
    int result = strcmp(str1, str2);
    if (result == 0) {
        printf("Strings are equal.\n");
    } else if (result < 0) {
        printf("str1 is less than str2.\n");
    } else {
        printf("str1 is greater than str2.\n");
    }
    return 0;
}
```

Output:
str1 is less than str2.

7. strstr() - Substring Search
Definition: Finds the first occurrence of a substring in a string.

```
#include <stdio.h>
#include <string.h>
int main() {
    char str[] = "Programming in C";
    char substr[] = "in";
    char *result = strstr(str, substr);
    if (result) {
        printf("Substring found: %s\n", result);
    } else {
        printf("Substring not found.\n");
    }
    return 0;
```

```
}
```

Output:
Substring found: in C

8. strchr() - Character Search
Definition: Finds the first occurrence of a character in a string.

```
#include <stdio.h>
#include <string.h>
int main() {
    char str[] = "Hello, World!";
    char ch = 'W';
    char *result = strchr(str, ch);
    if (result) {
        printf("Character found at position: %ld\n", result - str);
    } else {
        printf("Character not found.\n");
    }
    return 0;
}
```

Output:
Character found at position: 7

9. strrev() - String Reverse (Non-standard)
Definition: Reverses a string (available in some compilers like Turbo C, but not part of the standard library).

```
#include <stdio.h>
#include <string.h>
int main() {
    char str[] = "C Language";
    strrev(str);
    printf("Reversed String: %s\n", str);
    return 0;
}
```

Output (if supported):
Reversed String: egaugnaL C

10. strdup() - Duplicate String
Definition: Returns a pointer to a new string which is a duplicate of the original string.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int main() {
```

```
    char str[] = "Duplicate me";
    char *duplicate = strdup(str);
    if (duplicate) {
        printf("Duplicated String: %s\n", duplicate);
        free(duplicate);
    }
    return 0;
}
```

Output:
Duplicated String: Duplicate me

6. Common Mistakes with Strings
Forgetting to include the null character ('\0') when manually initializing strings.
Using scanf for reading strings with spaces.
Modifying string literals (e.g., str2[0] = 'A'; when str2 is declared using a pointer).

Important Questions on Strings in C

1. What is the difference between declaring a string using a character array and a pointer?
2. How do you reverse a string without using a library function?
3. Write a program to find the length of a string without using strlen().
4. Explain the use of strcpy and potential pitfalls when using it.
5. How would you concatenate two strings without using strcat()?
6. Write a program to check if a given string is a palindrome.
7. What is the purpose of the null character '\0' in strings?