Day 14 – Strings in C

🎯 Learning Objective

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

- Understand what a string is in C
- Declare and initialize strings
- Understand the relationship between strings and character arrays
- Understand the `'\0'` null character
- Read and display strings
- Find string length
- Reverse a string
- Check whether a string is a palindrome
- Use basic string-handling functions

1. What is a String?

A string is a sequence of characters terminated by a special character called the null character `'\0'`.

Example:

```
char name[] = "Bhavya";
```

Internally, C stores it as:

```
 B    h    a    v    y    a   \0
 ↓    ↓    ↓    ↓    ↓    ↓    ↓
[0]  [1]  [2]  [3]  [4]  [5]  [6]
```

Important point

Although `"Bhavya"` contains **6 visible characters**, the array needs **7 positions** because C automatically adds `'\0'` at the end.

2. String Declaration

You should learn these two forms:

Method 1 – Character array

```
char name[20];
```

This creates space for up to 19 characters plus `'\0'`.

Method 2 – Initialization

```
char name[] = "Bhavya";
```

The compiler automatically creates enough space for:

```
B h a v y a \0
```

3. First Program – Print a String

Start with the simplest possible example.

```
#include <stdio.h>

int main()
{
    char name[] = "Bhavya";

    printf("Student Name: %s\n", name);

    return 0;
}
```

Output

```
Student Name: Bhavya
```

Explain `%s`

You should remember:

```
%c  → one character
%s  → a string
```

Example:

```
char ch = 'A';
char name[] = "Bhavya";

printf("%c", ch);
printf("%s", name);
```

---

4. Reading a String from the User

Recommended Function: `fgets()`

Use `fgets()` as the main learning example.

```
#include <stdio.h>

int main()
{
    char name[50];

    printf("Enter your name: ");
    fgets(name, sizeof(name), stdin);

    printf("Welcome, %s", name);

    return 0;
}
```

Example

```
Enter your name: Bhavya
Welcome, Bhavya
```

Explain

```
fgets(name, sizeof(name), stdin);
```

Break it into three parts:

```
name
 ↓
Where should the input be stored?

sizeof(name)
 ↓
How much space is available?

stdin
 ↓
Take input from the keyboard
```

This is much better for beginners than simply memorizing the function.

---

5. `puts()` – Display a String

`puts()` is a simple function for displaying a string.

```
#include <stdio.h>

int main()
{
    char name[] = "Bhavya";

    puts(name);

    return 0;
}
```

Output:

```
Bhavya
```

Compare

```
printf("%s", name);
```

and

```
puts(name);
```

For beginners:

`printf("%s", name)` prints a string, while `puts(name)` prints a string and automatically moves to the next line.

6. What About `gets()`?

The syllabus mentions:

`gets()`

I recommend learning it only as a **historical concept**, not as a programming practice.

Explain to you:

`gets()` was traditionally used to read a string, but it is unsafe because it does not know the size of the destination array. It can cause buffer overflow. Therefore, modern C programs should use `fgets()` instead.

Simple comparison

```
gets()   → ✗ Unsafe
fgets()  → ✓ Safer and recommended
```

This is an important programming habit for you.

---

7. String Length

Now introduce the first useful string-processing problem.

Problem

Find the length of a string.

Example:

```
Bhavya
```

Length:

```
6
```

But remember:

```
B h a v y a \0
```

The `'\0'` is **not counted** as part of the string length.

Program 1 – Length Using `strlen()`

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

int main()
{
    char name[] = "Bhavya";

    printf("String: %s\n", name);
    printf("Length: %zu\n", strlen(name));

    return 0;
}
```

Output:

```
String: Bhavya
Length: 6
```

Explain

`strlen(name)`

counts the characters **before** `'\0'`.

8. Important Program – Find Length Without `strlen()`

For programming you, this is even more valuable because it develops logic.

```
#include <stdio.h>

int main()
{
    char name[50];
    int length = 0;

    printf("Enter a string: ");
    fgets(name, sizeof(name), stdin);

    while (name[length] != '\0' && name[length] != '\n')
    {
        length++;
    }

    printf("Length = %d\n", length);

    return 0;
}
```

Logic

```
Start
  ↓
length = 0
  ↓
Check character
  ↓
```

```
Is it '\0' or '\n'?
  ↓
No → length++
  ↓
Repeat
  ↓
Yes
  ↓
Display length
```

This is an excellent program for learning **arrays + loops + strings together**.

9. Reverse a String

Now move from basic string handling to logic.

Example

```
Original:

BHAVYA

Reverse:

AYVAHB
```

Program – Reverse a String

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

int main()
{
    char str[50];
    int i, length;

    printf("Enter a string: ");
    fgets(str, sizeof(str), stdin);

    length = strlen(str);

    if (str[length - 1] == '\n')
    {
        str[length - 1] = '\0';
        length--;
    }

    printf("Reversed string: ");

    for (i = length - 1; i >= 0; i--)
    {
        printf("%c", str[i]);
    }

    return 0;
}
```

Example

```
Enter a string: Bhavya

Reversed string: ayvahB
```

10. Challenge – Palindrome String

Now give you a challenge.

What is a Palindrome?

A palindrome reads the same from left to right and right to left.

Examples:

```
MADAM
LEVEL
RADAR
MALAYALAM
```

Non-palindromes:

```
BHAVYA
COMPUTER
PROGRAM
```

---

11. Palindrome Logic

For:

```
MADAM
```

Compare:

```
M ⟷ M
A ⟷ A
D ⟷ D
```

All characters match.

Therefore:

```
MADAM is a palindrome
```

12. Best Beginner Palindrome Program

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

int main()
{
    char str[100];
    int left, right;
    int palindrome = 1;

    printf("Enter a string: ");
    fgets(str, sizeof(str), stdin);

    int length = strlen(str);

    if (str[length - 1] == '\n')
    {
        str[length - 1] = '\0';
        length--;
    }

    left = 0;
    right = length - 1;

    while (left < right)
    {
        if (str[left] != str[right])
        {
            palindrome = 0;
            break;
        }

        left++;
        right--;
    }

    if (palindrome == 1)
        printf("Palindrome string\n");
    else
        printf("Not a palindrome string\n");

    return 0;
}
```

Example 1

```
Enter a string: MADAM

Palindrome string
```

Example 2

```
Enter a string: BHAVYA

Not a palindrome string
```

13. Visual Logic for You

For MADAM:

```
        M A D A M
        ↑       ↑
        |       |
       left    right

Compare M and M
       ↓
     Same

        M A D A M
          ↑   ↑
          |   |
        left right

Compare A and A
       ↓
     Same

        M A D A M
            ↑
            D

Only the middle character remains.

Therefore:

       PALINDROME ✅
```

Strings

A. String Fundamentals

1. What is a string?
2. Character vs String
3. Character array
4. String declaration
5. String initialization
6. `'\0'` null character
7. `%c` vs `%s`

B. String Input & Output

8. `printf()`
9. `puts()`
10. `fgets()`
11. Why `gets()` is unsafe

C. String Programs

12. Print a string
13. Read a string
14. Find string length using `strlen()`
15. Find length without `strlen()`
16. Reverse a string
17. Count characters

D. Challenge Programs

18. Palindrome string
19. Count vowels and consonants
20. Count spaces and digits

E. Introduction to String Library

Introduce:

```
strlen()
strcpy()
strcat()
strcmp()
```

Don't go deeply into all four on Day 14 if you are beginners. **Day 14 should primarily establish the string concept and logic.**

☆ Best Learning Sequence

For you, I recommend this progression:

```
Character
   ↓
Character Array
   ↓
String
   ↓
'\0'
   ↓
%s
   ↓
Input using fgets()
   ↓
Output using puts()
   ↓
Length
   ↓
Reverse
   ↓
Palindrome
```

This is much stronger pedagogically than simply learning:

```
gets()
puts()
fgets()
Length
Reverse
Palindrome
```

because you first understand **how a string actually exists in memory**, and then they learn how to process it.

One important correction

C Strings – Simple Programs with Examples

1. `puts()` – Display a String

Program

```
#include <stdio.h>

int main()
{
    char name[] = "Bhavya";

    puts(name);

    return 0;
}
```

Output

```
Bhavya
```

Explain to you

```
puts(name);
```

means: Display the complete string stored in `name`.

Remember

```
printf("%s", name);  → displays a string
puts(name);          → displays a string and moves to the next line
```

2. `fgets()` – Read a String from Keyboard

This is the **recommended method** for learning string input.

Program

```
#include <stdio.h>

int main()
{
    char name[50];

    printf("Enter your name: ");
    fgets(name, sizeof(name), stdin);

    printf("Your name is: ");
    puts(name);

    return 0;
}
```

Example Output

```
Enter your name: Bhavya
Your name is: Bhavya
```

Explain

```
fgets(name, sizeof(name), stdin);
```

means:

- `name` → store the input here
- `sizeof(name)` → maximum available size
- `stdin` → read from keyboard

Important

`fgets()` can read a complete line, including spaces.

Example:

```
Enter your name: Bhavya Raju
Your name is: Bhavya Raju
```

3. `gets()` – Old Method

You may show this program **only to explain the historical function**.

```
#include <stdio.h>

int main()
{
    char name[50];

    printf("Enter your name: ");
    gets(name);

    printf("Your name is: ");
    puts(name);

    return 0;
}
```

Example

```
Enter your name: Bhavya Raju
Your name is: Bhavya Raju
```

⚠ Important Learning Point

`gets()` is unsafe because it does not check the size of the character array. It can cause a buffer overflow and was removed from the C standard.

Therefore:

```
gets()   ✗ Do not use in new programs
fgets()  ☑ Use this
```

For modern C programming, learn **`fgets()` instead of `gets()`.**

4. Find String Length Using `strlen()`

Program

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

```
int main()
{
    char name[] = "Bhavya";

    printf("String: %s\n", name);
    printf("Length: %zu\n", strlen(name));

    return 0;
}
```

Output

```
String: Bhavya
Length: 6
```

Visual Explanation

```
B   h   a   v   y   a   \0
↓   ↓   ↓   ↓   ↓   ↓
1   2   3   4   5   6
```

The `'\0'` is the **null character**. It marks the end of the string and is **not counted** by `strlen()`.

5. Find Length Without `strlen()`

This is an excellent program for developing programming logic.

Program

```
#include <stdio.h>

int main()
{
    char name[50];
    int i = 0;

    printf("Enter a string: ");
    fgets(name, sizeof(name), stdin);

    while (name[i] != '\0' && name[i] != '\n')
    {
        i++;
    }

    printf("Length = %d\n", i);

    return 0;
}
```

Example

```
Enter a string: Computer

Length = 8
```

Logic

```
Start
  ↓
i = 0
  ↓
Check name[i]
  ↓
Is it \0 or \n?
  ↓ No
```

```
i++
  ↓
Repeat
  ↓
Yes
  ↓
Display length
```

6. Reverse a String

Example

```
Original:
BHAVYA

Reverse:
AYVAHB
```

Program

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

int main()
{
    char str[50];
    int i, length;

    printf("Enter a string: ");
    fgets(str, sizeof(str), stdin);

    length = strlen(str);

    if (str[length - 1] == '\n')
    {
        str[length - 1] = '\0';
        length--;
    }

    printf("Original String: ");
    puts(str);

    printf("Reverse String: ");

    for (i = length - 1; i >= 0; i--)
    {
        printf("%c", str[i]);
    }

    printf("\n");

    return 0;
}
```

Output

```
Enter a string: BHAVYA

Original String: BHAVYA

Reverse String: AYVAHB
```

The logic.

For:

```
B H A V Y A
0 1 2 3 4 5
```

Start from the last position:

```
5 → A
4 → Y
3 → V
2 → A
1 → H
0 → B
```

Therefore:

```
AYVAHB
```

7. Palindrome String

A **palindrome** is a string that remains the same when reversed.

Examples

```
MADAM   → MADAM  ✅
LEVEL   → LEVEL  ✅
RADAR   → RADAR  ✅
```

But:

```
BHAVYA  → AYVAHB  ❌
```

8. Simple Palindrome Program

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

int main()
{
    char str[50];
    int left, right;
    int palindrome = 1;

    printf("Enter a string: ");
    fgets(str, sizeof(str), stdin);

    int length = strlen(str);
```

```
    if (str[length - 1] == '\n')
    {
        str[length - 1] = '\0';
        length--;
    }

    left = 0;
    right = length - 1;

    while (left < right)
    {
        if (str[left] != str[right])
        {
            palindrome = 0;
            break;
        }

        left++;
        right--;
    }

    if (palindrome == 1)
        printf("Palindrome String\n");
    else
        printf("Not a Palindrome String\n");

    return 0;
}
```

Example 1

```
Enter a string: MADAM

Palindrome String
```

Example 2

```
Enter a string: COMPUTER

Not a Palindrome String
```

9. Visualize the Palindrome Logic

For: M A D A M
Compare the first and last characters:
M → ← M
They are equal.
Then:
A → ← A
They are equal.
The middle character is:
D
All comparisons are successful.
Therefore:
MADAM
↓
Palindrome ✅

☆ Recommended Practice Set for You

No.	Program	Main Concept
1	Display string using `puts()`	String output
2	Read string using `fgets()`	String input
3	Demonstrate `gets()`	Historical/unsafe method
4	Find length using `strlen()`	String library
5	Find length without `strlen()`	Loop + string
6	Reverse a string	Array indexing
7	Check palindrome	String logic
8	Count vowels	Character checking
9	Count consonants	Character checking
10	Count spaces	String processing

Simple Practice Programs

1. Display a String Using `puts()`

Concept

String output

Program

```
#include <stdio.h>

int main()
{
    char name[] = "Bhavya";

    puts(name);

    return 0;
}
```

Output

```
Bhavya
```

Key point

```
puts(name);
```

displays the complete string.

2. Read a String Using `fgets()`

Concept

String input

```
#include <stdio.h>

int main()
{
    char name[50];

    printf("Enter your name: ");
    fgets(name, sizeof(name), stdin);

    printf("Your name is: ");
    puts(name);

    return 0;
}
```

Output

```
Enter your name: Bhavya
Your name is: Bhavya
```

Key point

```
fgets(name, sizeof(name), stdin);
```

reads a line of text safely into the character array.

3. Demonstrate `gets()` — Historical Example

Concept

Understanding an unsafe function

⚠ `gets()` is unsafe and should **not** be used in modern C programs. Show this only to explain why `fgets()` is preferred.

```
#include <stdio.h>

int main()
{
    char name[50];

    printf("Enter your name: ");
    gets(name);

    printf("Your name is: ");
    puts(name);

    return 0;
}
```

Example

```
Enter your name: Bhavya Raju

Your name is: Bhavya Raju
```

Tell you

`gets()` → ✗ Unsafe

`fgets()` → ✅ Recommended

`gets()` does not know the size of the destination array and can cause a buffer overflow.

4. Find String Length Using `strlen()`

Concept

String library function

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

int main()
{
    char name[] = "Bhavya";

    printf("String = %s\n", name);
    printf("Length = %zu\n", strlen(name));

    return 0;
}
```

Output

```
String = Bhavya
Length = 6
```

Remember

```
B h a v y a \0
```

`strlen()` counts the characters but **does not count** `'\0'`.

5. Find String Length Without `strlen()`

Concept

Loop + string

This is a very important logic-building program.

```
#include <stdio.h>

int main()
{
    char str[50];
    int i = 0;

    printf("Enter a string: ");
    fgets(str, sizeof(str), stdin);

    while (str[i] != '\0' && str[i] != '\n')
    {
        i++;
    }

    printf("Length = %d\n", i);

    return 0;
}
```

Example

```
Enter a string: Computer

Length = 8
```

Logic

```
Start
  ↓
i = 0
  ↓
Check str[i]
  ↓
Character found?
  ↓ Yes
i++
  ↓
Repeat
  ↓
'\0' or '\n'?
  ↓
Stop
```

6. Reverse a String

Concept

Array indexing

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

int main()
{
    char str[50];
    int i, length;

    printf("Enter a string: ");
    fgets(str, sizeof(str), stdin);

    length = strlen(str);

    if (str[length - 1] == '\n')
    {
        str[length - 1] = '\0';
        length--;
    }

    printf("Reverse = ");

    for (i = length - 1; i >= 0; i--)
    {
        printf("%c", str[i]);
    }

    return 0;
}
```

Output

```
Enter a string: Bhavya
Reverse = ayvahB
```

Understand

```
B h a v y a
0 1 2 3 4 5

Start from 5
↓
a y v a h B
```

7. Check Palindrome

Concept

String logic

A palindrome reads the same forward and backward.

Examples:

```
MADAM  → Palindrome
LEVEL  → Palindrome
RADAR  → Palindrome
```

Program

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

int main()
{
    char str[50];
    int i, length;
    int palindrome = 1;

    printf("Enter a string: ");
    fgets(str, sizeof(str), stdin);

    length = strlen(str);

    if (str[length - 1] == '\n')
    {
        str[length - 1] = '\0';
        length--;
    }

    for (i = 0; i < length / 2; i++)
    {
        if (str[i] != str[length - 1 - i])
        {
            palindrome = 0;
            break;
        }
    }

    if (palindrome == 1)
        printf("Palindrome\n");
    else
        printf("Not a Palindrome\n");

    return 0;
}
```

Example

```
Enter a string: MADAM
Palindrome
```

Another example:

```
Enter a string: COMPUTER
Not a Palindrome
```

Visual

```
M A D A M
↑       ↑
M       M  ✓

  A   A
  ↑   ↑
  A   A    ✓
```

8. Count Vowels

Concept

Character checking

Vowels are:

```
A E I O U
a e i o u
```

Program

```
#include <stdio.h>

int main()
{
    char str[100];
    int i, vowels = 0;

    printf("Enter a string: ");
    fgets(str, sizeof(str), stdin);

    for (i = 0; str[i] != '\0'; i++)
    {
        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++;
        }
    }

    printf("Number of vowels = %d\n", vowels);

    return 0;
}
```

Example

```
Enter a string: Bhavya

Number of vowels = 2
```

The vowels are:

```
a
a
```

9. Count Consonants

Concept

Character checking + condition

A consonant is an alphabet character that is **not a vowel**.

Program

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

int main()
{
    char str[100];
    int i, consonants = 0;

    printf("Enter a string: ");
    fgets(str, sizeof(str), stdin);

    for (i = 0; str[i] != '\0'; i++)
    {
        if (isalpha((unsigned char)str[i]))
        {
            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')
            {
                consonants++;
            }
        }
    }

    printf("Number of consonants = %d\n", consonants);

    return 0;
}
```

Example

```
Enter a string: Bhavya

Number of consonants = 4
```

The consonants are:

```
B
h
v
y
```

10. Count Spaces

Concept

String processing

Program

```
#include <stdio.h>

int main()
{
    char str[100];
    int i, spaces = 0;

    printf("Enter a sentence: ");
    fgets(str, sizeof(str), stdin);

    for (i = 0; str[i] != '\0'; i++)
    {
        if (str[i] == ' ')
        {
            spaces++;
        }
    }

    printf("Number of spaces = %d\n", spaces);

    return 0;
}
```

Example

```
Enter a sentence: I love C programming

Number of spaces = 3
```

Visual:

```
I[ ]love[ ]C[ ]programming
   ↑       ↑  ↑
   1       2  3
```

☆ Recommended Learning Order

This creates a natural progression:

Input/Output → String Library → Loops → Array Indexing → Logic → Character Processing