



# Strings

---

Session 10



# Objectives

---

- Explain string variables and constants
- Explain pointers to strings
- Perform string input/output operations
- Explain the various string functions
- Explain how arrays can be passed as arguments to functions
- Describe how strings can be used as function arguments



# String variables

---

- **Strings are arrays of characters terminated by the NULL ('\0') character.**
- **String variables can be assigned string constants.**
- **A string constant is a sequence of characters surrounded by double quotes.**
- **The '\0' null character is automatically added in the internal representation of a string.**
- **While declaring a string variable, allow one extra element space for the null terminator.**



# Declaring string variables

---

- A typical string variable declaration is:.

**char str[10];**

- **str** is a character array variable that can hold a maximum of 10 characters including the null terminator.



# String I/O operations-1

---

- String I/O operations are carried out using functions from the standard I/O library called **stdio.h**
- The gets() function is the simplest method of accepting a string through standard input
- Input characters are accepted till the Enter key is pressed
- The gets() function replaces the terminating '\n' new line character with the '\0' character
- Syntax :

**gets(str);**



# String I/O operations-2

---

- The puts() function is used to display a string on the standard output device.
- Syntax :

**puts(str);**

- The scanf() and printf() functions are used to accept and display mixed data types with a single statement.
- The syntax to accept a string is as follows:

**scanf("%s", str);**

- The syntax to display a string is as follows:

**printf("%s", str);**



# String Functions

---

Functions for handling strings are found in the standard header file **string.h**. Few of the operations performed by these functions are:

- Concatenating strings
- Comparing strings
- Locating a character in a string
- Copying one string to another
- Calculating the length of a string



# The strcat() function

---

- Joins two string values into one.
- Syntax:

**strcat(str1, str2);**

- Concatenates the str2 at the end of str1
- The function returns str1





# The strcmp() function

---

- Compares two strings and returns an integer value based on the results of the comparison.

- Syntax:

**strcmp(str1, str2);**

- The function returns a value:
  - Less than zero if  $\text{str1} < \text{str2}$
  - Zero if str1 is same as str2
  - Greater than zero if  $\text{str1} > \text{str2}$



# The strchr() function

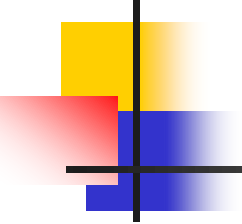
---

- Determines the occurrence of a character in a string.

- Syntax:

**strchr(str, chr);**

- The function returns a value:
  - Pointer to the first occurrence of the character (pointed by **chr**) in the string, **str**
  - NULL if it is not present



# The strcpy() function

---

- Copies the value in one string onto another

- Syntax:

**strcpy(str1, str2);**

- The value of str2 is copied onto str1
- The function returns **str1**



# The `strlen()` function

---

- Determines the length of a string

- Syntax:

**`strlen(str);`**

- The function returns an integer value for the length of **`str`**



# Passing Arrays to Functions-1

---

- When an array is passed as an argument to a function, only the address of the array is passed
- The array name without the subscripts refers to the address of the array

```
void main()  
{  
    int ary[10];  
    .  
    .  
    fn_ary(ary);  
    .  
    .  
}
```



# Passing Arrays to Functions-2

---

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

```
void main()
```

```
{
```

```
int num[5], ctr, sum=0;
```

```
int sum_arr(int num_arr[]); /* Function declaration */
```

```
clrscr();
```

```
for(ctr=0;ctr<5;ctr++) /* Accepts numbers into the array */
```

```
{
```

```
    printf("\nEnter number %d: ", ctr+1);
```

```
    scanf("%d", &num[ctr]);
```

```
}
```



# Passing Arrays to Functions-3

```
sum=sum_arr(num); /* Invokes the function */  
  
printf("\nThe sum of the array is %d", sum);  
  
getch();  
}  
  
int sum_arr(int num_arr[]) /* Function definition */  
{  
    int i, total;  
  
    for(i=0,total=0;i<5;i++) /* Calculates the sum */  
        total+=num_arr[i];  
  
    return total; /* Returns the sum to main() */  
}
```



# Passing Arrays to Functions-4

---

Sample output of the program

Enter number 1: 5

Enter number 2: 10

Enter number 3: 13

Enter number 4: 26

Enter number 5: 21

The sum of the array is 75





# Example of Passing Strings to Functions-1

---

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

```
void main()
{
char lines[5][20];
int ctr, longctr=0;
int longest(char lines_arr[][20]);
/* Function declaration */

clrscr();
for(ctr=0;ctr<5;ctr++)
/* Accepts string values into the array */
{
    printf("\nEnter string %d: ", ctr+1);
    scanf("%s", lines[ctr]);
}
```

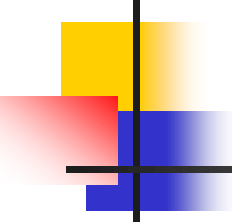


# Example of Passing Strings to Functions-2

---

```
longctr=longest(lines);  
/* Passes the array to the function */  
  
printf("\nThe longest string is %s", lines[longctr]);  
  
getch();  
}  
  
int longest(char lines_arr[][20]) /* Function definition */  
{  
    int i=0, l_ctr=0, prev_len, new_len;  
  
    prev_len=strlen(lines_arr[i]);  
    /* Determines the length of the first element */
```

# Example of Passing Strings to Functions-3



```
for(i++;i<5;i++)
{
    new_len=strlen(lines_arr[i]);
    /* Determines the length of the next element */

    if(new_len>prev_len)
        l_ctr=i;
    /* Stores the subscript of the longer string */

    prev_len=new_len;
}

return l_ctr;
/* Returns the subscript of the longest string */
}
```



# Example of Passing Strings to Functions-4

---

Sample output of the program

Enter string 1: The

Enter string 2: Sigma

Enter string 3: Protocol

Enter string 4: Robert

Enter string 5: Ludlum

The longest string is Protocol