STRING

What is the definition of string ? Explain with example. **OR**

Explain String in C with Example. **OR**

What is character array in C. Explain with suitable example **OR**

How to Declare and Initialize string in C ? Give example **OR**

Ans:

A string is a sequence of characters that is treated as a single data item. In C a string is represented as a array of characters arranged at contiguous block of memory where each element represent single character. Null Character ('\0') is used to indicate the end of string.

The general form of **declaration** of a string variable is:

char string_name[size];

The **size** determines the number of characters in the string_name.

char name[15]; //string or character array of size 15

Initialization of String

C permits a character array to be initialized in either of the following two forms

char city [9] = " NEW YORK ";

char city [9]={'N','E','W',' ','Y','O','R','K','\0'};

Here

city is an array of characters

"NEW YORK" is stored in this array.

city includes null character ('\0') at the end, making it a valid string.

Simple program to demonstrate the use of string using input/output function.

```
#include<stdio.h>
int main()
{
    char name[30];
    printf("Name :");
    scanf("%s",name);
    printf("Your input string is : %s",name);
    return 0;
}
```

e.g. Input : DONALD

Output : Your input string is : DONALD

STRING

How to read a string from the terminal ?

The input function scanf can be used with %s format specification to read in a string of characters.

```
char address[10];
scanf("%s", address);
```

scanf function terminates its input on the first white space it finds.

What is the limitations of scanf function while reading a string from the terminal ?

The input function scanf can be used with %s format specification to read in a string of characters.

```
char address[10];
scanf("%s", address);
```

scanf function terminates its input on the first white space it finds.

A white space includes blanks, tabs, carriage returns, form feeds, and new lines. Therefore, if the following line of text is typed in at the terminal, NEW YORK then only the string "NEW" will be read into the array address, since the blank space after the word 'NEW' will terminate the reading of string.

Note that unlike scanf calls for integer or fload variables, in the case of character arrays, the ampersand (&) is not required before the variable name.

How to declare and initialize the string without declaring size of an array ? Explain with example.

C permits to initialize a character array without specifying the number of elements.

In such cases, the **size of the array will be determined automatically**, based on the number of elements initialized.

For example, the statement

char string[ ] = {'G','O','O','D','\0'};

defines the array string as a five element array.

STRING

How to overcome the limitations of scanf function's reading string from the terminal ? OR

What is edit set conversion code in C ? OR

The input function scanf can be used with %s format specification to read in a string of characters.

Example: **char address[10];**

scanf("%s", address);

The problem with the scanf function is that it terminates its input on the first white space it finds. Therefore, if the following line of text is typed in at the terminal, NEW YORK then only the string "NEW" will be read into the array address, since the blank space after the word 'NEW' will terminate the reading of string.

Format specifier **%s** or **%ws** can read only strings without whitespaces. It cannot be used for reading a text containing more than one word.

To overcome this limitations C supports a format specification known as the **edit set conversion code** **"%[^\n]"** that can be used to read a line containing a variety of characters, including whitespaces.

For example, the program segment

```
char line [80];
scanf("%[^\n]", line);
printf("%s", line);
```

Will read a line of input including white space, from the keyboard and display the same on the screen.

Explain getchar() function ? OR

How to read a single character from the terminal ? OR

C getchar is a standard library function that takes a single input character from standard input.

```
#include<stdio.h>
int main(){
    char ch;                          //declare character variable ch
    ch = getchar( );                  //read single character
    printf("character read = %c",ch); //print character on screen
    return 0;
}
```

This small program reads single character using getchar() function and print on screen using printf() function.

STRING

Explain gets() function with simple example.

gets(str) is a library function reads a string of text containing whitespaces. This is a simple function with one string parameter and called as under: **gets (str);**

```
#include<stdio.h>
void main()
{
        char str[50];
        gets(str);
        printf("%s",str);
}
```

It reads characters into str from the keyboard until a new-line character is encountered and then appends a null character to the string. Unlike scanf, it does not skip whitespaces.

Write a program to read a line of text containing a series of words from the terminal.

```
#include <stdio.h>
int main( )
{
        char line[81], character;
        int c; c = 0;
        printf("Enter text. Press at end\n");
        do {
                character = getchar();
                line[c] = character; c++;
                } while(character != '\n');
                c = c - 1;
                line[c] = '\0';
                printf("\n%s\n",line);
        return 0;
}
```

In the above program, we can read a line of text (up to a maximum of 80 characters) into the string line using getchar function. Every time a character is read, it is assigned to its location in the string line and then tested for newline character. When the newline character is read (signalling the end of line), the reading loop is terminated and the newline character is replaced by the null character to indicate the end of character string.

STRING

Write a program to copy one string into another and count the number of characters copied

```
#include<stdio.h>
int main( )
 {
 char string1[80], string2[80];
 int i;
 printf("Enter a string \n");
 printf("?");
 scanf("%s", string2);
 for( i=0 ; string2[i] != '\0'; i++)
 string1[i] = string2[i];
 string1[i] = '\0';
 printf("\n");
 printf("%s\n", string1);
 printf("Number of characters = %d\n", i );
 return 0;
 }
```

STRING

Q. How to write string to screen ?

String can be written on string using printf() function or using putchar() or puts() function.

Using printf function :

We use the printf function with %s format to print strings to the screen. The format %s can be used to display an array of characters that is terminated by the null character.
For example, the statement

printf("%s", name);

can be used to display the entire contents of the array name. We can also specify the precision with which the array is displayed.
For instance, the specification
%10.4

indicates that the first four characters are to be printed in a field width of 10 columns.

```
#include<stdio.h>

int main()
{
	char country[15] = "United Kingdom";
	//  printf("\n\n");
	printf("*123456789012345*\n");
	printf(" – – - - - - - - \n");
	printf("%s\n", country);
	printf("%15s\n", country);
	printf("%15.6s\n", country);
	printf("%-15.6s\n", country);
	printf("%15.0s\n", country);
	printf("%.3s\n", country);
	printf("%s\n", country);
	printf("——--- \n");
	printf("%*.*s\n", -17, 5, country);
	return 0;
}
```

```
*123456789012345*
– – - - -
United Kingdom
United Kingdom
United
United
 Uni
 United Kingdom
  –  –  -  -  -
```

STRING

Using putchar and puts function

putchar function

The putchar(int ch) method in C is used to write a character, of unsigned char type, to stdout. This character is passed as the parameter to this method.

Syntax

```
int putchar(int ch)
```

Parameters: This method accepts a mandatory parameter ch which is the character to be written to stdout.

Return Value: This function returns the character written on the stdout as an unsigned char. It also returns EOF when some error occurs.

```c
#include <stdio.h>
int main()
{
    // Get the character to be written
    char ch = 'G';
    // Write the Character to stdout
    putchar(ch);
    return (0);
}
```

STRING

puts function

puts() is a function defined in header **<stdio.h>** that prints strings character by character until the NULL character is encountered. The puts() function prints the newline character at the end of the output string.

Syntax

```
int puts(char* str);
```

Parameters

- **str:** string to be printed.

Return Value

The return value of the puts function depends on the success/failure of its execution.

- On success, the puts() function returns a non-negative value.
- Otherwise, an End-Of-File (EOF) error is returned.

```
// C program to illutrate the use of puts() function
#include <stdio.h>
int main()
{
    // using puts to print hello world
    char* str1 = "Hello Geeks";
    puts(str1);
    puts("Welcome Geeks");
    return 0;
}
```

Describe the ARITHMETIC OPERATIONS ON CHARACTERS.

C allows us to manipulate characters the same way we do with numbers. Whenever a character constant or character variable is used in an expression, it is automatically converted into an integer value by the system. The integer value depends on the local character set of the system.

To write a character in its integer representation, we may write it as an integer.

For example, if the machine uses the ASCII representation,

then, x = ‘a’;
printf(“%d\n”,x);

will display the number 97 on the screen.

To perform arithmetic operations on the character constants and variables.

For example,
x = ‘z’–1;
is a valid statement
Here ASCII value of ‘z’ is 122 hence 122 – 1 = 121 will be assigned to ‘x’ variable

We can use character constants in relational expressions like ch >= ‘A’ && ch <= ‘Z’
would test whether the character contained in the variable ch is an upper-case letter

STRING

Write a program which would print the alphabet set a to z and A to Z in decimal and character form.

```
#include<stdio.h>
main()
{
	char c;
	printf(“\n\n”);
	for( c = 65 ; c <= 122 ; c = c + 1 )
	{
		if( c > 90 && c < 97)
		continue;
		printf(“|%4d - %c “, c, c);
	}
	printf(“|\n”);
	return 0;
}
```

Output:

| 65 - A | 66 - B | 67 - C | 68 - D | 69 - E | 70 - F | 71 - G | 72 - H | 73 - I | 74 - J | 75 - K | 76 - L | 77 - M| 78 - N| 79 - O| 80 - P| 81 - Q| 82 - R | 83 - S| 84 - T| 85 - U| 86 - V| 87 - W| 88 - X | 89 - Y| 90 - Z| 97 - a| 98 - b| 99 - c| 100 - d |101 - e| 102 - f| 103 - g| 104 - h| 105 - i| 106 - j | 107 - k| 108 - l| 109 - m| 110 - n| 111 - o| 112 - p | 113 - q| 114 - r| 115 - s| 116 - t| 117 - u| 118 - v | 119 - w| 120 - x| 121 - y| 122 - z|

STRING

Concatenate String

What is Concatenation of string ? How to concatenate two strings together ? Explain with Example.

The process of combining two strings together is called concatenation.

Program to concatenate two strings.

```
#include<stdio.h>
#include<string.h>
int main()
{
    char str1[100] = "hello" ,str2[100] = " world";
    int i,j;
    i = strlen(str1);
    for(j=0;str2[j]!='\0';i++,j++)
    {
        str1[i] = str2[j];          //assign each character of str2 to str1
    }
    str1[i] = '\0';
    printf("Concatenated string is: %s",str1);
    return 0;
}
```

COMPARISON OF TWO STRINGS

Explain comparison of two string without using standard lib function strcat .

C does not permit the comparison of two strings directly. That is, the statements such as if(name1 == name2) if(name == “ABC”) are not permitted. It is therefore necessary to compare the two strings to be tested, character by character.
The following segment of a program illustrates this.

```
#include<stdio.h>
#include<string.h>
/*program to compare two strings without using strcmp*/
int main()
{
      char str1[100] = "hello", str2[100] = "hello";
      int i,j;

      for(i=0,j=0;str1[i]!='\0' && str2[j]!='\0';i++,j++)
      {
         if(str1[i]!=str2[j])
         break;
      }
      if(str1[i]==str2[j])
        printf("Strings are Equal");
      else
        printf("String are Not Equal");
      return 0;
}
```

STRING

STRING-HANDLING FUNCTIONS

FUNCTION	ACTION
strcat	Concatenates two strings
strcmp	Compares two strings
strcpy	Copies one string over another
strlen	Finds the length of a string

strcat() function:

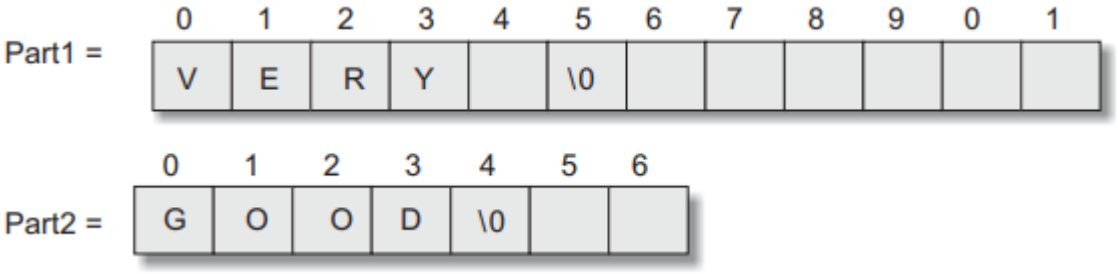
The **strcat** function joins two strings together.

It takes the following form:

strcat(string1, string2);

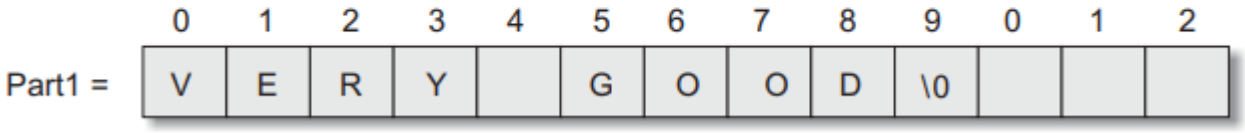
string1 and **string2** are character arrays.

When the function strcat is executed, string2 is appended to string1. It does so by **removing the null character at the end of string1** and **placing string2** from there. The string at string2 remains unchanged. For example, consider the following three strings




```
strcat(part1, part2);
```

will result in:




```
#include<stdio.h>
#include<string.h>
//program to concatenate two strings using strcat()
int main()
{
   char str1[100] = "hello", str2[100] = " world";
   strcat(str1,str2);
   printf("Concatenated string is: %s",str1);
   return 0;
}
```

STRING

Describe strcmp function in c? Give simple example to demonstrate the use of strcmp function.

The **strcmp** function compares two strings identified by the arguments and has a value 0 if they are equal.

The strcmp() function returns three different values after the comparison of the two strings which are as follows:

- Zero (0): It returns zero when all of the characters at given indexes in both strings are the same.
- Greater than Zero (> 0): Returns a value greater than zero is returned when the first not-matching character in s1 has a greater ASCII value than the corresponding character in s2.
- Lesser than Zero (< 0): If the value is negative, s1 is alphabetically above s2.

```
#include <stdio.h>
#include <string.h>
int main(){
        char s1[10] = "hello";
        char s2[10] = "hello";

        int l = strcmp(s1,s2);
        if(l == 0)
        {
                printf(”strings are equal”);
        }
        else
        {
                printf(“strings are not equal”);
        }

        printf("%d", strcmp(s1, s1));

   return 0;
}
```

STRING

Explain strcpy function. **OR**
What is the use of strcpy function and demonstratre it's use ? **OR**
Explain in short about strcpy function.

strcpy() Function

The function prototype of `strcpy()` is:

```
char* strcpy(char* destination, const char* source);
```

- The `strcpy()` function copies the string pointed by `source` (including the null character) to the destination.
- The `strcpy()` function also returns the copied string.

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

int main() {
  char str1[20] = "C programming";
  char str2[20];

  // copying str1 to str2
  strcpy(str2, str1);

  puts(str2); // C programming

  return 0;
}
```

Output

```
C programming
```

STRING

strlen() Function

This function counts and returns the number of characters in a string.

It takes the form n = strlen(string);

Where n is an integer variable, which receives the value of the length of the string. The argument may be a string constant. The counting ends at the fi rst null character

```
#include<stdio.h>
#include<string.h>
int main()
{
        char str1[30] = “hello world”;
        int length = strlen(str1);
        printf(“length of str1 = %d”,length);
        return 0;
}
```

Output : length of str1 = 11;

i.e. There are 11 characters are present in the string str1;

strncmp() Function

The **`strncmp`** function compares at most a specified number of characters from two strings in alphabetical order.

This function has three parameters as illustrated in the function call below:

strncmp (s1, s2, n);

This compares the left-most n characters of s1 to s2 and returns.

(a) 0 if they are equal;
(b) negative number, if s1 sub-string is less than s2; and
(c) positive number, otherwise.

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

int main() {
   char str1[] = "apple";
   char str2[] = "apricot";

   // Compare the first 3 characters
   int result = strncmp(str1, str2, 3);

   if (result == 0) {
      printf("The first 3 characters are equal.\n");
   } else if (result < 0) {
      printf("'%s' is less than '%s' in the first 3 characters.\n", str1, str2);
   } else {
      printf("'%s' is greater than '%s' in the first 3 characters.\n", str1, str2);
   }
   return 0;
}
```

STRING

strncpy() Function

Function **strncpy** that copies only the left-most n characters of the source string to the target string variable.

This is a three parameter function and is invoked as follows:

strncpy(s1, s2, 5);

This statement copies the first 5 characters of the source string s2 into the target string s1. Since the first 5 characters may not include the terminating null character, we have to place it explicitly in the 6th position of s2 as shown below: s1[6] ='\0'; Now, the string s1 contains a proper string.

```
#include <stdio.h>
#include <string.h>
int main() {
   char src[] = "Hello, World!";
   char dest[20]; // Ensure the destination buffer is large enough

   // Copy the first 5 characters of src to dest
   strncpy(dest, src, 5);

   // Explicitly null-terminate dest
   dest[5] = '\0';

   printf("Source: %s\n", src);
   printf("Destination: %s\n", dest);

   return 0;
}
```

strstr() Function

strstr It is a two-parameter function that can be used to locate a sub-string in a string. This takes the following forms:

```
strstr (s1, s2);
strstr (s1, "ABC");
```

The function strstr searches the string s1 to see whether the string s2 is contained in s1. If yes, the function returns the position of the first occurrence of the sub-string. Otherwise, it returns a NULL pointer.

Example:

```
if (strstr (s1, s2) == NULL)
        printf("substring is not found");
else
        printf("s2 is a substring of s1");
```

strchr() Function

The **strchr** function is used to find the first occurrence of a character in a string. It searches through a string and returns a pointer to the first occurrence of the specified character, or `NULL` if the character is not found.

```
strchr(s1, 'm');
```

will locate the first occurrence of the character 'm'

STRING

strrchr() Function

The `strrchr` function is used to locate the **last occurrence** of a character in a string. It searches a string from the end towards the beginning and returns a pointer to the last occurrence of the specified character, or `NULL` if the character is not found.

strrchr(s1, ‘m’); will locate the last occurrence of the character ‘m’ in the string s1

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

STRING

Function	Description	Example Statement
`strcat`	Concatenates (appends) one string to the end of another.	`strcat(dest, src);` Example: If `dest = "Hello "` and `src = "World"`, result: `dest = "Hello World"`.
`strcmp`	Compares two strings lexicographically. Returns 0 if equal, < 0 if the first is less, > 0 if greater.	`strcmp("apple", "banana");` Returns: `< 0` because `"apple"` is less than `"banana"`.
`strlen`	Returns the length of the string (excluding the null terminator).	`strlen("Hello");` Returns: `5`.
`strcpy`	Copies the source string into the destination string (including the null terminator).	`strcpy(dest, src);` If `src = "Hello"`, then `dest = "Hello"`.
`strncpy`	Copies up to `n` characters from the source string to the destination string.	`strncpy(dest, src, 3);` If `src = "Hello"`, then `dest = "Hel"`.
`strncmp`	Compares up to `n` characters of two strings lexicographically.	`strncmp("apple", "apricot", 3);` Returns: `0` because the first 3 characters are the same.
`strncat`	Appends up to `n` characters from the source string to the destination string.	`strncat(dest, src, 3);` If `dest = "Hi "` and `src = "there"`, result: `dest = "Hi the"`.
`strstr`	Finds the first occurrence of a substring in a string. Returns a pointer to the start of the substring.	`strstr("Hello World", "World");` Returns: Pointer to `"World"`.

STRING

Function Name	Description	Example Statement
strcat(dest, src)	Concatenates (appends) the source string (`src`) to the end of the destination string (`dest`).	`strcat(dest, " world!");` (if `dest` initially holds "Hello,")
strcmp(str1, str2)	Compares two strings lexicographically. Returns 0 if they are equal, a negative value if `str1` is less than `str2` , and a positive value if `str1` is greater than `str2` .	`strcmp("apple", "banana")` (returns a negative value)
strlen(str)	Returns the length of the string (number of characters before the null character).	`strlen("hello")` (returns 5)
strcpy(dest, src)	Copies the source string (`src`) to the destination string (`dest`).	`strcpy(dest, "Hello");`
strncpy(dest, src, n)	Copies at most `n` characters from the source string (`src`) to the destination string (`dest`).	`strncpy(dest, "Hello", 3);` (copies "Hel" to `dest`)
strncmp(str1, str2, n)	Compares at most `n` characters of two strings. Returns 0 if they are equal up to `n` characters, a negative value if `str1` is less than `str2` , and a positive value if `str1` is greater than `str2` .	`strncmp("hello", "help", 3)` (returns 0)
strncat(dest, src, n)	Concatenates at most `n` characters from the source string (`src`) to the end of the destination string (`dest`).	`strncat(dest, " world!", 5);` (appends " worl" to `dest`)
strstr(str1, str2)	Finds the first occurrence of the substring `str2` within the string `str1` . Returns a pointer to the first occurrence of `str2` in `str1` , or NULL if `str2` is not found.	`strstr("hello world", "world")` (returns a pointer to the first occurrence of "world")