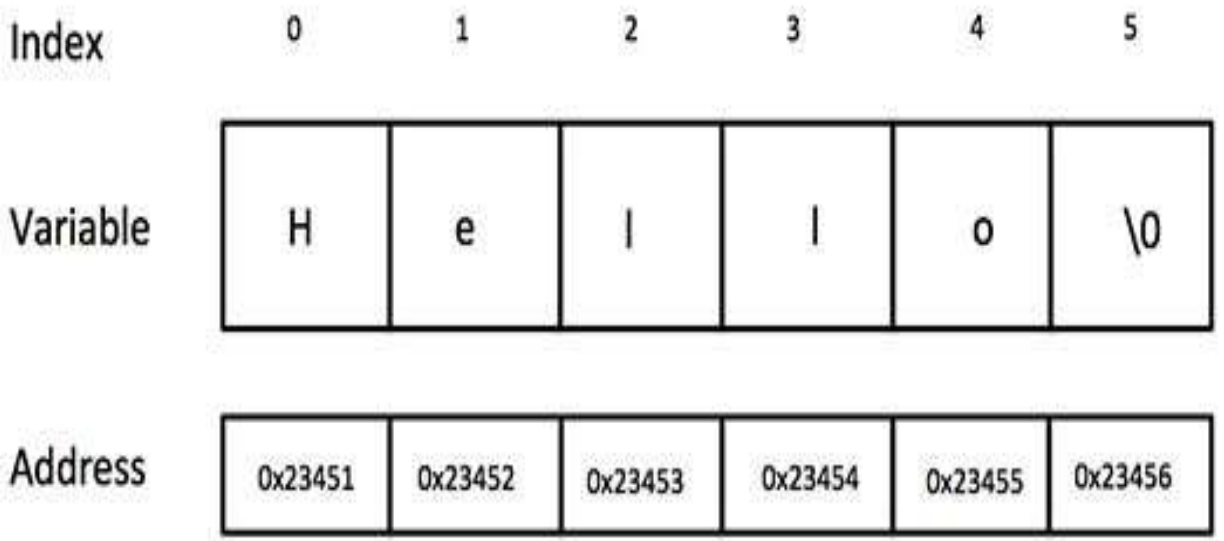
String

- String is a sequence of characters that are treated as a single data item and terminated by a null character '\0'.
- The string can be defined as the one-dimensional array of characters.
- Each character in the array occupies one byte of memory, and the last character must always be 0.
- The C language does not support strings as a data type.

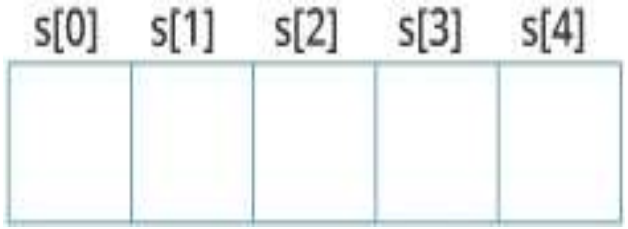
Example


Index
0
1
2
3
4
5
Variable
H
e
l
l
o
\0
Address
0x23451
0x23452
0x23453
0x23454
0x23455
0x23456

Declaring and Initializing a string variables

- Declaring a string variables

char s[5];



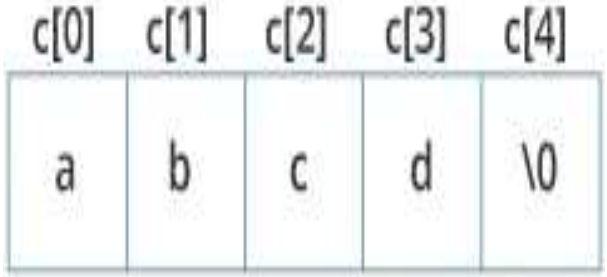
- Initializing a string variables

char c[] = "abcd";

char c[50] = "abcd";

char c[] = {'a', 'b', 'c', 'd', '\0'};

char c[5] = {'a', 'b', 'c', 'd', '\0'};



String Example in C

```
 #include<stdio.h>
#include <string.h>
int main(){
  char ch[10]={'O', 'S', 'W', '\0'};
   char ch1[10]="OSW";

   printf("%s\n", ch);
   printf("%s\n", ch1);
 return 0;
}
```

Array of Strings

- A string is a 1-D array of characters, so an array of strings is a 2-D array of characters.
- We can create a 2-D array of int, float etc; we can also create a 2-D array of character or array of strings.
- Declare a 2-D array of characters.

```
char ch_arr[3][10] = {
                        {'s', 'p', 'i', 'k', 'e', '\0'},
                        {'t', 'o', 'm','\0'},
                        {'j', 'e', 'r', 'r', 'y','\0'}
                     };
```

Example..

```
#include<stdio.h>
int main()
{ int i;
char ch_arr[3][10] = {
                        "spike",
                        "tom",
                        "jerry"
                      };
    printf("1st way \n\n");

    for(i = 0; i < 3; i++)  {
        printf("string = %s \t address = %u\n", ch_arr + i, ch_arr + i);
    }
    return 0; }
```

Gets() & puts() function

- Both the functions are used to in the input and output operation of the Strings

- The gets() functions are used to read string input from the keyboard and puts() function displays it.

- These functions are declared in the stdio.h header file.

gets() function

- The gets() function is similar to scanf() function but it allows entering some characters by the user in string format with the double quotation. and stored in a character array format.

puts() function

- The puts() function is similar to printf() function. but puts() function is used to display only the string after reading by gets() function entered by user.

Example

```
 #include<stdio.h>
#include <string.h>
int main(){
char name[50];
printf("Enter your Subject name: ");
gets(name); //reads string from user
printf("Your Subject name is: ");
puts(name);  //displays string
return 0;
}
```

fgets()

- For reading a string value with spaces, we can use fgets() in C programming language.
- It reads a line from the specified stream and stores it into the string pointed to by str. It stops when either (n-1) characters are read, the newline character is read, or the end-of-file is reached, whichever comes first.
- Since fgets() reads input from user, we need to provide input during runtime.
- Syntax
- **char*** fgets(**char** *s, **int** n, **stdin**)
- The fgets() function returns a **pointer** to the string buffer if successful. A **NULL** return value indicates an error or an end-of-file condition.

Example:

```
#include<stdio.h>
int main(){
char str[20] ;
printf(“Enter the string”);
fgets(str, 20, stdin);
printf(“The entered string is: %s\n”, str );
return 0;}
```

Example

```
#include<stdio.h>
int main(){
char str[20];

fputs("Hello world", stdout);

return 0;}
```

String Functions

Function	Description
strlen(string_name)	returns the length of string name.
strcpy(destination, source)	copies the contents of source string to destination string.
strcat(first_string, second_string)	concats or joins first string with second string. The result of the string is stored in first string.
strcmp(first_string, second_string)	compares the first string with second string. If both strings are same, it returns 0.
strrev(string)	returns reverse string.
strlwr(string)	returns string characters in lowercase.
strupr(string)	returns string characters in uppercase.

strlen() function

- The strlen() function returns the length of the given string. It doesn't count null character '\0’.

- The return value is of type size_t (an unsigned integer type)

- Syntax:

  size_t strlen(const char *str)

  str : string whose length is to be found

Example

```
#include<stdio.h>
#include <string.h>
int main(){
char ch[20]={'O', 'S', 'W', '\0'};
printf("Length of string is: %ld\n",strlen(ch));
return 0;
 }
```

Output: 3

strcpy()

- The strcpy(destination, source) function copies the source string in destination.

```
#include<stdio.h>
#include <string.h>
int main(){
char ch[20]={'O', 'S', 'W', '\0'};
char ch1[20];
strcpy(ch1,ch);
printf("Value of second string is: %s",ch1);
 return 0;
}
```

strtok()

Declaration:

char *strtok(char *str, const char *delim);

It breaks string **str** into a series of tokens using the delimiter **delim**.

- Return Value

- This function returns a pointer to the first token found in the string.

- A null pointer is returned if there are no tokens left to retrieve.

Example

```
#include<stdio.h>
#include<string.h>
int main(){
char str[40] = "This is a book";
char *token;
token = strtok(str, " ");
while(token != NULL){
        printf("%s\n", token);
        token = strtok(NULL, " ");
}
return 0;}
```

strtok r() function

```
#include<stdio.h>
#include<string.h>
int main() {
char str[] ="Lesson-plan-USP-DOS-FML-PLC";
printf("Entered strin:::");
puts(str);
char *token;
char *last;
token = strtok_r(str, "-", &last);
while (token!=NULL) {
 printf("Token:%s\n", token);
printf("\t\tRemaining part of the string:%s\n",last);
token = strtok_r(NULL, "-", &last); }
return (0); }
```

strcat()

- The strcat(first_string, second_string) function concatenates two strings and result is returned to first_string.

```
#include<stdio.h>
#include <string.h>
int main(){
char ch[20]={'O', 'S', 'W', '\0'};
char ch1[10]={'C', 'L', 'A', 'S', 'S', '\0'};
strcat(ch,ch1);
printf("Value of first string is: %s",ch);
 return 0;
}
```

Compare String: strcmp()

- The function strcmp is of int type.
- It compares two strings that we will refer to as str1 and str2 .
- Function strcmp separates its argument pairs into three categories as follows:

Relationship	Value Returned
str1 is less than str2	negative integer
str1 equals str2	zero
str1 is greater than str2	positive integer

Possible Results of strcmp(str1, str2)

Return value : 0, if two strings are equal

negative, if str1 < str2

positive, if str1>str2

No other assumptions should be made about the value returned by **strcmp**.

- If the first n characters of str1 and str2 match and str1[n] , str2[n] are the first nonmatching corresponding characters,

  str1 is less than str2, if str1[n] < str2[n] .

  Example:

str1 t h r i l l	str1 e n e r g y
str2 t h r o w	str2 f o r c e
First 3 letters match.	First 0 letters match.
str1[3] < str2[3]	str1[0] < str2[0]
'i' < 'o’	'e' < ‘f’

- If str1 is shorter than str2 and all the characters of str1 match the corresponding characters of str2 , str1 is less than str2 .

  str1 j o y

  str2 j o y o u s

strncmp()

- It bases its comparison on only the first n characters of the two strings, where n is the third argument.
- Syntax:
- strncmp(str1, str2, n);
- Example:
- str1 = "joyful" str2 = "joyous“

strncmp(str1, str2, 1)

strncmp(str1, str2, 2)

strncmp(str1, str2, 3)

strncmp(str1, str2, 4)

Example

```
 #include<stdio.h>
 #include <string.h>
int main(){
char str1[20],str2[20];

printf("Enter 1st string: ");
scanf(“%s”, str1);//reads string from console
printf("Enter 2nd string: ");
 scanf(“%s”, str2);
if(strcmp(str1,str2)==0)
    printf("Strings are equal");
else
      printf("Strings are not equal");
 return 0;  }
```

Reverse String: strrev()

- The strrev(string) function returns reverse of the given string.

```
#include<stdio.h>
#include <string.h>
int main(){
  char chr[20];
  printf("Enter string: ");
  gets(chr); //reads string from console
  printf("String is: %s",chr);
  printf("\nReverse String is: %s",strrev(chr));
 return 0;
}
```

- strrev() is non-standard and may not be available.

String Lowercase: strlwr()

- The strlwr(string) function returns string characters in lowercase.

```
#include<stdio.h>
#include <string.h>
int main(){
  char str[20];
  printf("Enter string: ");
  gets(str);//reads string from console
  printf("String is: %s",str);
  printf("\nLower String is: %s",strlwr(str));
 return 0;
}
```

- strlwr() is non-standard and may not be available.

String Uppercase: strupr()

- The strupr(string) function returns string characters in uppercase.

```
#include<stdio.h>
#include <string.h>
int main(){
  char str[20];
  printf("Enter string: ");
  gets(str); //reads string from console
  printf("String is: %s",str);
  printf("\nUpper String is: %s",strupr(str));
 return 0;
}
```

- strupr() is non-standard and may not be available.

Character Analysis and Conversion

- In many string-processing applications, we need to know if a character belongs to a particular subset of the overall character set.
- Is this character a letter? a digit? A punctuation mark?
- The library we #include as <ctype.h> defines facilities for answering questions like these and also provides routines to do common character conversions like uppercase to lowercase or lowercase to uppercase.

Character Classification and Conversion Facilities in ctype Library

Facility	Checks
isalpha	if argument is a letter of the alphabet
isdigit	if argument is one of the ten decimal digits
islower (isupper)	if argument is a lowercase (or uppercase) letter of the alphabet
ispunct	if argument is a punctuation character, that is, a noncontrol character that is not a space, a letter of the alphabet, or a digit
isspace	if argument is a whitespace character such as a space, a newline, or a tab
tolower (toupper)	its lowercase (or uppercase) letter argument to the uppercase (or lowercase) equivalent and returns this equivalent as the value of the call

String-to-Number and Number-to-String Conversions

- Some of the most common operations in a computer program are the conversion of a string like "3.14159" to the type double numeric value it represents and the conversion of a number like −36 from its internal representation in computer memory to the three-character string "−36" that is our usual picture of this number.
- Such conversions are constantly being carried out by the library functions scanf and printf .

Use of scanf

Declaration	Statement	Data (␣ means blank)	Value Stored
`char t`	`scanf("%c", &t);`	`␣g`	
		`\n`	`\n`
		`A`	`A`
`int n`	`scanf("%d", &n);`	`␣32␣`	`32`
		`␣␣-8.6`	`-8`
		`␣+19␣`	`19`
`double x`	`scanf("%lf", &x);`	`␣␣␣4.32␣`	`4.32`
		`␣-8␣`	`-8.0`
		`␣1.76e-3␣`	`.00176`

Used with printf

Value	Placeholder	Output (▮ means blank)
'a'	%c	a
	%3c	▮▮a
	%-3c	a▮▮
-10	%d	-10
	%2d	-10
	%4d	▮-10
	%-5d	-10▮▮
49.76	%.3f	49.760
	%.1f	49.8
	%10.2f	▮▮▮▮49.76
	%10.3e	▮4.976e+01
"fantastic"	%s	fantastic
	%6s	fantastic
	%12s	▮▮▮fantastic
	%-12s	fantastic▮▮▮
	%3.3s	fan

The %3.3s placeholder indicates output of a string using a minimum field width of 3 (**3**.3) and a maximum field width of 3 (3.**3**). As a result, only the first three characters of the string are printed.