**13. STRINGS**

**13.1 Introduction:**

A string is a sequence of characters. A string is commonly enclosed within double quotes. Example: “Chennai”

A NULL character (‘\0’) is automatically appended by the compiler at the end of every string.

Consider a string “Chennai”.

**Compiler View: User View:**

**“Chennai\0” “Chennai”**

The NULL character is not usually considered while calculating the length of the string.

The number of bytes required to store a string constant is one more than the number of characters.

An empty string is denoted by “”. It takes one byte to store a NULL character.

White (Blank) spaces are included while calculating the size of the string usingsizeof().

**13.2 Difference between string and character.**

|  |  |
| --- | --- |
| String | Character |
| A sequence of characters. | A single character. |
| Strings are enclosed within double quotes (“”).  Example: “Book” | A character is enclosed within single quotes (‘’).  Example: ‘a’ |
| Memory allocation is one byte more than that of the character. | Memory allocation is one byte. |
| White (Blank) space are included. | White (Blank) space are included but only one space are allowed, because maximum of size is one byte. |
| Empty string is allowed.  Example: (“”) | Empty char is not allowed.  Example: (‘’) |

**13.3 STRING HANDLING:**

In C language, the collection of characters, digits and symbols are enclosed within quotation marks are called strings. The string is always declared as character arrays. String is terminated with ‘\0’ (NULL) character.

**13.4 Declaration and initialization:**

C does not support string as a data type. So they are declared as an array of characters.

Syntax for declaration:

**char string\_name[size]**;

**Display of strings with different formats:**

Let the character array be

Char name[]=”computer”;

|  |  |  |  |
| --- | --- | --- | --- |
| S.No | Statement | Output | Meaning |
| 1. | printf(“%s”,name); | Computer | The whole string is displayed |
| 2. | printf(“%.5s”,name); | Compu | Only 5 characters are displayed |
| 3. | printf(“%-10.4s”,name); | Comp | 4 characters from the left are displayed |

Example:

char a[10];

Syntax for initialization:

**char string\_name[size]**=”string\_array”**;**

**Example:**

**char a[10] =”reading”;**

**Explanation:**

Here a character array of size 10 is initialized, so values up to 10 characters can be accepted .Here “reading” has occupied has occupied 7 bytes of storage and in the end they are terminated with a NULL (\0).

**Alternate syntax for initialization:**

**char string\_name[]**=”string\_array”**;**

**char string\_name[]**={‘char’,’\0’}**;**

**Example:**

**char a[]={“Chennai”};**

**Note:**

\*C also permits to initialize a character array without specifying the number of elements

**\*NULL (\0) character is accepted by the compiler only when the initialization is done character by character.**

.

Example:

char b[]={'a',’c’,'\0'};

In the absence of a particular array size, the C compiler automatically calculates the number of elements in the array based upon the number of the characters initialized.

Program:

#include <stdio.h>

#include<conio.h>

int main()

{

char a[6]="hello";

char b[]="hi";

char c[]={'h','a','i','\0'};

clrscr();

printf("\n %s \n %s \n %s ",a,b,c);

return 0;

}

Output:

hello

hi

hai

**13.4.1 Reading strings:**

The user can enter strings and store them in character arrays at the run time.

Different ways to read a string:

* Format specifier
* Field width
* Search set

Format specifier:

Instead of using ‘%c’ as a specifier to get a string character by character, the format specifier ‘%s’ can be used in a scanf statement to get a string from the user.

scanf() function reads a character array only upto a white (blank) space.

Sample program:

#include<stdio.h>

#include<conio.h>

void main()

{

char name[20];

clrscr();

printf(“\n Enter a name:\t”);

scanf(“%s”,name);

printf(“\n The name is: %s“,name);

getch();

}

Output 1:

Enter a name: Abc

The name is: Abc

Output 2:

Enter a name: Chennai city

The name is: Chennai

Explanation:

From output2, it is clear that in scanf() function, space and enter is considered as termination.

**13.4.2 Field width**:

The scanf() function can be used to read a specific number of characters by mentioning the field width.

Sample program:

#include<stdio.h>

#include<conio.h>

void main()

{

char name[20];

clrscr();

printf(“\n Enter a name:\t”);

scanf(“%5s”,name);

printf(“\n The name is: %s“,name);

getch();

}

Output:

Enter a name: Ramanujam

The name is: Raman

Explanation:

In the above example the field width is mentioned as 5 hence the scanf() function can recognize only the first five characters.

**13.4.3 Search set**:

The scanf function can also be used to read selected characters by making use of search sets.

A search set defines a set of possible characters that can make up the string.

A search set is enclosed within square brackets ‘[]

Sample program:

#include<stdio.h>

#include<conio.h>

void main()

{

char name[20];

clrscr();

printf(“\n Enter a name:\t”);

scanf(“%[abcde]”,name);

printf(“\n The name is: %s“,name);

getch();

}

Output:

Enter a name: apple

The name is: ae

Explanation:

In the above example search set is given as [abcde] hence the scanf functions accepts only those characters therefore ‘ae’ is printed in the output.

**13.4.4 Writing strings**:

The printf function with ’ %s ‘ format specification is used to print strings to the screen.

The format ‘%s’ can be used to display an array of characters that is terminated by the NULL (\0) character.

**Advantage**:

Two or more strings can be printed by a single call to the function having multiple ‘%s’ specifiers.

**Sample program:**

#include<stdio.h>

#inlcude<conio.h>

Void main()

{

char name[5];

clrscr();

printf(“\n Enter a name:\t”);

scanf(“%s”,name);

printf(“\n Name is %s ”,name);

printf(“\n Enter a name again:\t”);

scanf(“%s”,name);

printf(“\n Name is %s ”,name);

getch();

}

Output:

Enter a name: karguvel

The name is: karguvel

Enter a name again: karguvel

The name is karguvel

Examples for Reading strings:

|  |  |  |  |
| --- | --- | --- | --- |
| S.No | Statement | Output | Meaning |
| 1. | printf(“%s”,name); | Computer | The whole string is displayed |
| 2. | printf(“%.5s”,name); | Compu | Only 5 characters are displayed |
| 3. | printf(“%+10.4s”,name); | Comp | ‘.4’ refers 4 characters from the left are displayed  ‘+10’ specifies the space from the left |

**13.5 Processing the strings:**

The strings can be processed either by using some predefined functions with the help of ‘string.h’ header file or by processing all characters individually.

**13.6 STRING STANDARD FUNCTIONS**

C Language supports a large number of string handling library functions. Some of the standard string functions are given below:

|  |  |  |
| --- | --- | --- |
| **S.No** | **Function** | **Description** |
| 1. | strlen() | Determines the length of the string |
| 2. | strcpy() | Copies a string from source to destination |
| 3. | strcmp() | Compares the characters of two strings. |
| 4. | strcat() | Appends source string to destination string |
| 5. | strrev() | Reverses the string |
| 6. | **strchr(str, ch);** | Returns a pointer to the first occurrence of character ch in string str. |
| 7. | **strstr(s1, s2);** | Returns a pointer to the first occurrence of string s2 in string s1. |
| 8. | **strncat** | concatenate one string with part of another. |
| 9. | **strncmp** | compare parts of two strings. |
| 10. | **strncpy** | copy part of a string |
| 11. | **Strrchr** | string scanning operation |

**Example** −For strcpy,strcat,strlen

#include <stdio.h>

#include <string.h>

int main ()

{

char str1[12] = "SJIT";

char str2[12] = "CSE";

char str3[12];

int len ;

strcpy(str3, str1);

printf("strcpy( str3, str1) : %s\n", str3 );

strcat( str1, str2);

printf("strcat( str1, str2): %s\n", str1 );

len = strlen(str1);

printf("strlen(str1) : %d\n", len );

return 0;

}

Output:

strcpy( str3, str1) : SJIT

strcat( str1, str2): SJIT CSE

strlen(str1) : 7

**13.6.1 strncat** function:

The function char \*strncat(char \*dest, const char \*src, size\_t n) appends the string pointed by source to end of the string pointed by destination up to n characters long.

**Syntax:**

char \*strncat(char \*dest, const char \*src, size\_t n)

**Parameters**

dest -- This is the pointer to destination array, which should have a C string, and should be big enough to have the concatenated resulting string which also includes the additional null-character.

src -- This is the string to be appended.

n -- This is the maximum number of characters to be appended.

**Return Value**

This function returns a pointer to the resulting string destination.

**Example:**

**The following example shows the usage of strncat() function.**

#include <stdio.h>

#include <string.h>

int main ()

{

char s[50], d[50];

strcpy(s, "this is source");

strcpy(d, "This is destination");

strncat(d, s, 15);

printf("Final destination string : |%s|", d);

return(0);

}

**Output:**

Final destination string : |This is destination this is source|

**13.6.2 Strncmp function:**

The function int strncmp(const char \*s1, const char \*s2, size\_t n) is used to compare at most the first n bytes of s1 and s2.

**Syntax:**

int strncmp(const char \*s

1, const char \*str2, size\_t n)

**Parameters**

str1 -- This is the first string to be compared.

str2 -- This is the second string to be compared.

n -- the maximum number of characters to be compared.

**Return Value**

if return value < 0 then str1 is less than str2.

if Return value > 0 then str2 is less than str1.

if Return value = 0 then str1 is equal to str2.

**Example**

**The following example shows the usage of strncmp() function.**

#include <stdio.h>

#include <string.h>

int main ()

{

char str1[15];

char str2[15];

int ret;

strcpy(str1, "abcdef");

strcpy(str2, "ABCDEF");

ret = strncmp(str1, str2, 4);

if(ret < 0)

{

printf("str1 is less than str2");

}

else if(ret > 0)

{

printf("str2 is less than str1");

}

else

{

printf("str1 is equal to str2");

}

return(0);

}

Output:

str2 is less than str1

**13.6.3 Strncpy function:**

The strncpy() function copies a part of the contents of one string into another.

**Declaration:**

Following is the declaration for strncpy() function.

char \*strncpy(char \*dest, const char \*src, size\_t n)

**Parameters:**

dest -- This is the pointer to the destination array where the content is to be copied.

src -- This is the string to be copied.

n -- The number of characters to be copied from source.

**Return Value**

This function returns the final copy of the copied string.

Example for **strncpy**:

#include <stdio.h>

#include <string.h>

int main()

{

char src[40];

char dest[12];

memset(dest, '\0', sizeof(dest));

strcpy(src, "This is tutorialspoint.com");

strncpy(dest, src, 10);

printf("Final copied string : %s\n", dest);

return(0);

}

Output:

Final copied string : This is tu

**13.6.4 strrchr** function:

The function char \*strrchr(const char \*s, int c) searches the last occurrence of the character c (an unsigned char) in the string, by the argument s.

Syntax:

char \*strrchr(const char \*str, int c)

**Parameters:**

str -- This is the C string.

c -- This is the character to be located. It is passed as its int promotion, but it is internally converted back to char.

**Return Value**

It returns a pointer to the last occurrence of the character in str. A null pointer will be returned by the function if the value is not found.

Example for **strrchr**:

#include <stdio.h>

#include<string.h>

int main() {

char \*s;

char buf [] = "This is a testing";

s = strrchr (buf, 't');

if (s != NULL)

printf ("found a 't' at %s\n", s);

return 0;

}

Output:

found a 't' at ting

**13.6.5 Reversing the string**:

The strrev fuction reverses all the characters of a string except the terminating null character.

**Syntax:**

strrev(string);

**Example**:

#include<stdio.h>

#include<string.h>

#include<conio.h>

void main()

{

char s1[20],s2[20];

clrscr();

printf("\n enter the string:");

gets(s1);

strcpy(s2,s1);

strrev(s2);

printf("\n reversed string:%s",s2);

getch();

}

Output:

enter the string: PROGRAMMING

reversed string: GNIMMARGORP

**Example Program String operations using pre-defined functions**

#include <stdio.h>

#include <conio.h>

#include <string.h>

void str\_len();

void str\_comp();

void str\_con();

void str\_cpy();

char a[25],b[25],c[50];

void main()

{

int choice;

clrscr();

printf("1. finding the length of the string");

printf("\n2. string comparison");

printf("\n3. string copy");

printf("\n4. String concatenate");

printf("\nEnter ur choice");

scanf("%d",&choice);

switch(choice)

{

case 1:

str\_len();

break;

case 2:

str\_comp();

break;

case 3:

str\_cpy();

break;

case 4:

str\_con();

break;

default:

exit(1);

}

getch();

}

void str\_len()

{

int n;

fflush(stdin);

printf("\n Enter the string");

gets(a);

n=strlen(a);

printf("\nThe length of the string is %d",n);

}

void str\_comp()

{

fflush(stdin);

printf("\n Enter the I string");

gets(a);

printf("\nEnter the II String");

gets(b);

if(strcmp(a,b)==0)

printf("\n The two strings are identical");

else

printf("\nThe strings are different");

}

void str\_cpy()

{

fflush(stdin);

printf("\n Enter the string");

gets(a);

strcpy(b,a);

printf("\nThe copied string :");

puts(b);

}

void str\_con()

{

fflush(stdin);

printf("\n Enter the I string");

gets(a);

printf("\nEnter the II String");

gets(b);

strcat(a,b);

printf("\nThe concatenated string is : ");

puts(a);

}

**OUTPUT:**

1. finding the length of the string

2. string comparison

3. string copy

4. String concatenate

### Enter ur choice 1

Enter the string computer

The length of the string is 9.

**13.7 String operations without using pre-defined functions**

#include <stdio.h>

#include <conio.h>

#include <string.h>

void str\_len();

void str\_con();

void str\_cpy();

char a[25],b[25],c[50];

void main()

{

int choice;

clrscr();

printf("1. finding the length of the string");

printf("\n2. string copy");

printf("\n3. String concatenate");

printf("\nEnter ur choice");

scanf("%d",&choice);

switch(choice)

{

case 1:

str\_len();

break;

case 2:

str\_cpy();

break;

case 3:

str\_con();

break;

default:

exit(1);

}

getch();

}

void str\_len()

{

int n=0,i;

fflush(stdin);

printf("\n Enter the string");

gets(a);

for(i=0;a[i]!='\0';i++)

n++;

printf("\nThe length of the string is %d",n);

}

void str\_cpy()

{

int i;

fflush(stdin);

printf("\n Enter the string");

gets(a);

for(i=0;a[i]!='\0';i++)

b[i]=a[i];

printf("\nThe copied string :");

puts(b);

}

void str\_con()

{

int n=0,i;

fflush(stdin);

printf("\n Enter the I string");

gets(a);

printf("\nEnter the II String");

gets(b);

for(i=0;a[i]!='\0';i++)

{

n++;

}

for(i=0;b[i]!='\0';i++)

{

a[n++]=b[i];

}

printf("\nThe concatenated string is : ");

puts(a);

}

**OUTPUT:**

1. finding the length of the string

2. string copy

3. String concatenate

Enter ur choice 2

Enter the string computer

The copied string : computer.

**13.8 Program to sort names in alphabetical order**

#include <stdio.h>

#include <conio.h>

void main()

{

char a[25][25],i,j,n,temp[20];

clrscr();

printf("\nEnter the no. of strings in the array");

scanf("%d",&n);

printf("\nEnter the strings in the array\n");

for(i=0;i<n;i++)

{

scanf("%s",a[i]);

}

for(i=0;i<n-1;i++)

{

for(j=i+1;j<n;j++)

{

if(strcmp(a[i],a[j])>0)

{

strcpy(temp,a[i]);

strcpy(a[i],a[j]);

strcpy(a[j],temp);

}

}

}

printf("\nThe sorted strings in the array is \n");

for(i=0;i<n;i++)

{

printf("%s\n",a[i]);

}

getch();

}

**OUTPUT:**

Enter the no. of strings in the array3

Enter the strings in the array

cse

ece

it

The sorted strings in the array is

cse

ece

it