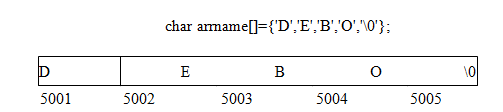
**UNIT-III**

**CHAPTER – 1**

**Strings**

A string is a sequence of characters that is treated as a single data item. Any group of characters defined between double quotation marks is a string constant.

Ex:char arrname[]=”DEBO”;



Usually, we have two types of strings and they are:

1.Fixed-length string

2.Variable length string

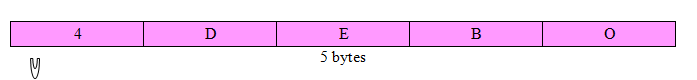
**Fixed-length string** is a type of string where length of string is fixed. No matter how many characters constitute the string, the space is fixed.Length of string cannot be changed, once defined. And spaces are usually added towards the end of string and they are apparently considered as non data characters. There are some disadvantages related to fixed length strings. For instance, if you have a large string and you have reserved smaller space then larger half of string will not be stored. In contrast, if you have a small string and you have reserved larger space then limited memory is wasted. Not only this, once defined, length of string cannot be changed. To solve these problems we have variable length strings.

**Variable-length string**is a type of string whose length is unknown as the name suggests. Storage container contracts and expands according to requirement of string length. But there has to be way to indicate the finish signal to compiler and for that purpose we have two general approaches and they are as follows:

1.Length controlled variable length string

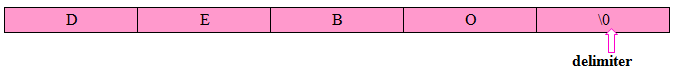
2.Delimited variable length string

When we talk about length controlled string then as the name suggests, length information of string is stored as part of string. This information is stored as first byte and this is used as a counter. We can have any length of string which is in between 0-255 range. For example, “debo” strings would be stored as follows:



**Figure – C Length controlled string**

In this scenario, there is one disadvantage and that is a if you have 4 byte length string as in the above example we will have one extra byte long string as length information is also a part of string so a 4 byte long string is actually going to occupy 5 bytes so this a sort of demerit when you have limited space and long strings to store. So we have one more option and that is delimited variable length which has a delimiter. Delimited string is the one which has NULL as the delimiter.



**C String:**

A C string is a variable-length array of characters that is delimited by the null character. C uses variable length and delimited string.

**Storing Strings:**

In C, a string is stored in an array of characters. It is terminated by the null character (‘\0’). The name of the string is a pointer to the beginning of the string.

**String Literal:**

A String Literal, also known as a string constant or constant string, is a string of characters enclosed in double quotes

Ex: “ Hai Hello “

Character strings are often used to build meaningful and readable programs. The common operations performed on character strings include:

* Reading and writing strings.
* Combining strings together.
* Copying one string to another.
* Comparing strings for equality.
* Extracting a portion of a string.

**Declaring and Initializing String Variable:**

C does not support strings as a data type. So, it allows us to represent strings as character arrays. A string variable is any valid C variable name and is always declared as an array of characters. 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.

Ex: char student[10];

char name[30];

When the compiler assigns a character string to a character array, it automatically supplies a null character (‘\0’) at the end of the string. Therefore, the size should be equal to the maximum number of characters in the string plus one.

Character arrays may be initialized when they are declared. We can do this by either of the following two forms:

Char name [30] = “ SRI DEVI”;

Char name [30] = {‘S’, ‘R’, ‘I’, ‘ ‘, ‘D’, ‘E’, ‘V’, ‘I’, ‘\0’};

When we are initializing a character array by listing its elements, we must supply explicitly the null terminator.

C also permits us to initialize a character array without specifying the number of elements. In such cases, the size of array will be determined automatically, based on the number of elements initialized.

Ex: char string [ ] = {‘T’, ‘A’, ‘B’, ‘L’, ‘E’, ‘\0’};

defines the array string as a six element array.

Ex: char str[10] = “GOOD”;

In the above case, the computer creates a character array of size 10, places the value “GOOD” in it, terminates with the null character, and initializes all other elements to NULL.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| G | O | O | D | \0 | \0 | \0 | \0 | \0 | \0 |

The following declarations are not allowed:

Ex: char str3[5];

Str3 = “GOOD”;

Ex: char s[3] = “GOOD”;

Ex: char s1[4] = “abc”;

char s2[4];

s2 = s1;

**Reading Strings From Terminal:**

Using Scanf Function :

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

Ex: char a[10];

scanf (“%s”, a);

The problem with the scanf function is that it terminates its input on the first white space it finds. A white space includes blanks, tabs and new lines.

Ex: NEW YORK

Then only the string “NEW” will be read into the array address, since the blank spaces after the word “NEW” will terminate the reading of string.

The address can be created in the memory as

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| N | E | W | \0 | \0 | \0 | \0 | \0 | \0 | \0 |

Ex: char adr1[5], adr2[5];

Scanf (“%s %s”,adr1,adr2);

With the line of text NEW YORK

Will assign the string “NEW” to adr1 and “YORK” to adr2.

Will can also specify the field width using the form %ws in the scanf statement for reading a specified number of characters from the input string.

Ex: scanf (“%ws”, name);

Here, two things may happen.

* The width w is equal to or greater than the number of characters typed in. The entire string will be stored in the string variable.
* The width is less than the number of characters in the string. The excess characters will be truncated and left unread.

Ex: char name[10];

Scanf (“%5s”,name);

The input string RAM will be stored in the memory as:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| R | A | M | \0 | \0 | \0 | \0 | \0 | \0 | \0 |

The input string KRISHNA will be stored in the memory as:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| K | R | I | S | H | \0 | \0 | \0 | \0 | \0 |

**String-Handling Functions:**

C library supports a large number of string-handling functions that can be used to carry out many of the string manipulations. Following are the most commonly used 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.

Syn: strcat (string1, string 2);

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.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| V | E | R | Y |  | \0 |  |  |  |  |  |

Ex: part1 =

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| G | O | O | D | \0 |  |  |  |

Part2 =

Execution of the statement

Strcat (part1, part2);

Will result in :

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| V | E | R | Y |  | G | O | O | D | \0 |  |

part1 =

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| G | O | O | D | \0 |  |  |  |

Part2 =

We must make sure that the size of string1 (to which string2 is appended) is large enough to accommodate the final string.

Strcat function may also append a string constant to a string variable.

Ex: strcat (part1, “GOOD”);

C permits nesting of strcat functions.

Ex: strcat (strcat (string1, string2), string3);

Is allowed and concatenates all the three strings together. The resultant string is stored in string1.

**Strcmp () Function:**

The strcmp function compares two strings identified by the arguments and has a value 0 if they are equal. If they are not equal, it has the numeric differences between the first nonmatching characters in the strings.

Syn: strcmp (string1, string 2);

String1 and string2 may be string variables or string constants.

Ex: strcmp (name1, name2);

strcmp (name1, “John”);

strcmp (“their”, “There”);

will return a value of -9 which is the numeric difference between ASCII “i” and ASCII “r”. String1 is alphabetically above string2.

**Strcpy () Function:**

The strcpy function works almost like a string-assignment operator.

Syn: strcpy (string1, string 2);

It assigns the contents of string2 to string1. String2 may be a character array variable or a string constant.

Ex: strcpy (name, “RAM”);

Will assign the string “RAM” to the string variable name

strcpy (student1, student2);

will assign the contents of the string variable student2 to the string variable student1. The size of the array student1 should be large enough to receive the contents of student2.

**Strlen () Function:**

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

Syn: 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 first null character.

**Other String Functions:**

The header file <string.h> contains many more string manipulation functions.

**Strncpy () Function:**

This function is used to copy the left-most n characters of the source string to the target string variable. This is a three parameter function.

Syn: strncpy (string1, string 2,n);

Ex: strncpy (a,b,n);

This statements copies the first 5 characters of the source string b into the target string a. Since the first 5 characters may not include the terminating null character, we have to place it explicitly in the 6th position of b as shown below:

B[6]=’\0’;

Now, the string a contains a proper string.

**Strncmp () Function:**

This function has three parameters as shown below:

Syn: strncmp (string1, string 2,n);

Ex: strncmp (a, b, n);

This compares the left-most n characters of a to b and returns.

(a) 0 if they are equal

(b) negative number, if a sub-string is less than b; and

(c) positive number, otherwise.

**Strncat () Function:**

This function takes three parameters as shown below:

Syn: strncat (string1, string 2,n);

This concatenas the left-most n characters of b to the end of a.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| H | A | I | \0 |  |  |  |  |  |

Ex: part1 =

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| H | E | L | L | O | \0 |  |  |

Part2 =

Execution of the statement

Strncat (part1, part2,4);

Will result in :

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| H | E | L | L | \0 |  |  |  |  |

part1 =

**Strstr () Function:**

This is a two parameter function that can be used to locate a sub-string in a sring.

Syn : strstr (string1,string2);

Ex: strstr (s1, s2);

Strstr (s1, “RAM”);

The function 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.

Ex: if (strstr (s1, s2) == NULL)

Printf (“Substring is not found”);

Else

Printf (“s2 is asubstring of s1”);

We also have functions to determine the existence of a character in a string.

Ex: strchr (s1,’m’);

Will locate the first occurrence of the character ‘m’ and

Ex: strrchr (s1,’m’);

Will locate the last occurrence of the character ‘m’ in the string s1.

**Table Of Strings:**

A list of names can be treated as a table of strings and a two dimensional character array can be used to store the entire list.

Ex: A character array student [30][15] may be used to store a list of 30 names, each of length not more than 15 characters.

Char city[ ][ ]

{

“Chandigarh”,

“Madras”,

“Ahmedabad”,

“Hyderabad”,

“Bombay”

};

To access the name of the ith city in the list, we write

City [i-1]

And therefore city[0] denotes “Chandigarh”, city[1] denotes “Madras” and so n.