Module - II

Advanced C Programming

2.5 Arrays and strings

Definition:

A string is a sequence of characters terminated with a null character {'\0'}

Declaring and Initializing String Variables:-

* C does not support string as a data type but it can be declared using character Array.
* Declaring a string is as simple as declaring a one or two dimensional Array.

Syntax:-

Declaring a string is as declaring one dimensional Array.

```
char String_Variable_name [array_size]
```

where,

char - Data type

String_Variable_name - User defined valiable name

array_size - constant consists of integer value as number of memory space for array variable.

Example:-

```
char str[10];
char str[];
```

Initializing String Variable:-

* The User can store size -1 character in Array
* The last character would be the null chara in string or character array.

Initialization methods

Method - 1 character Array

Ex:- Char str[6] = {'H', 'E', 'L', 'L', 'O', '\0'};

or

char str[] = {'H', 'E', 'L', 'L', 'O', '\0'};

Index →	str[0]	str[1]	str[2]	str[3]	str[4]	str[5]
Value →	H	E	L	L	O	\n
Address →	6000	6001	6002	6003	6004	6005

method - 2 String literal

Ex:- Char str[10] = {"Hello"};

Index →	str[0]	str[1]	str[2]	str[3]	str[4]	str[5]	str[6]	str[7]	str[8]	str[9]
Value →	H	E	L	L	O	\0	\0	\0	\0	\0
Address	6000	6001	6002	6003	6004	6005	6006	6007	6008	6009

Note:-

* String is always declared as character Array.
* String literal are enclosed with double quotes and character literal are enclosed with single quotes.
* The string is always ended with null character \0
* The characters after the null character are ignored
* The length of the characters/string defined as the numbers of characters present in it. The delimiting NULL character is not counted.
* A Empty string is a string withe zero length is called Empty String.

Advantages of Array:-

* Array can store more than one value at a time.
* Direct indexing is supported by arrays.
* Direct indexing means the time required to acce any element in an array of any dimension.

Disadvantages of Array:-

* The memory to an array is allocated at the compile time
* The size of the Array cannot be expanded as the size was static
* The Elements in the array must be same data type.
* Memory usage in case of array is inefficient.

Printing Strings:-

Strings can be displayed on screen using 3 ways.

1. Using printf() function
2. Using puts() function
3. Using putchar() function.

Reading Strings:-

Strings can be read by using 3 ways.

1. Using scanf() function.
2. Using gets() function.
3. Using getchar(), getch() or getch() function.

Example:-

scanf() function to read a string.

```
int main()
  {
    char name[20];
    printf("Enter name:");
    scanf("%s", name);
    printf("Your name is %s:", name);
    return 0;
  }
```

output:-

```
Enter name: Santhosh kumar.
Your name is santhosh.
```

In the above example Santhosh only printed eventhough entered as Santhosh kumar, its because space there after Santhosh.

Example :- fgets() and puts()

```
int main()
{
  char name[30];
  Printf("Enter name: ");
  fgets(name, sizeof(name), stdin);
  Printf("Name: ");
  return 0;
}
```

Output:

Enter name : Santhosh kumar

Name: Santhosh kumar.

Two Dimensional Array string:-

→ Declaring a string is as declaring a two dimensional Array.

Syntax:

```
char str_name[size][max]
```

eg:-

```
char str_arr[2][6] = {{'R', 'a', 'j', 'i', '\0'}
                      {'r', 'a', 'm', '\0'}};
```

or

```
Char str_arr[2][6] = {"Raji", "ram"};
```

index ROWS	0	1	2	3	4	5
0	R	a	j	i	\0	\0
1	r	a	m	\0	\0	\0

Example :-

```
int main
{
 int i
 char name[2][2];
for(i=0; i<2; i++)
{
 printf("Enter name %d", i);
 scanf("%s", name[i]);
}
for(i=0; i<2; i++)
{
printf("String = %s\n", name[i]);
}
return 0;
}
```

Output:-

Enter name 0 raja

Enter name 1 ram

String = raja

String = ram.

2-6 Functions and String :-

* C has many useful string functions, which can be used to perform certain Operations on Strings.

Function declaration to accept one dimensional string:-

* we saved the arrays as string. So, we can pass an one dimensional array to a function by using the following declaration.

Syntax:-

returntype functionName (char str[]);

Example

Void displayString (char str[]);

* In the above example, we have a function by name displayString and it takes an argument of type char and the argument is an one dimensional array str[].

Passing one dimensional string to a Function:-

* To pass a one dimensional string to a function as an argument, just write the name of the string array variable.

Example:-

```
int main (void)
{
char message[] = "Hello world";

displayString (message);
return 0;
}

void displayString (char str[])
{
  int i=0;
  printf ("String:");
  while (str[i] != '\0')
  {
    printf ("%c", str[i]);
    i++;
  }
  printf ("\n");
}
```

Output:

String: Hello world

Function Declaration to accept 2D String

* In order to accept 2D string array the function declaration will look like the following,

Syntax:-

```
returnType functionName (char [][], type rows);
```

Example:-

```
Void displayCities (char str[][50], int rows);
```

* In the above example, we have function by the name displaycities and it takes a two dimensional string array of type char.
* In the above example str is a 2D string array as because it have 2 [] brackets.
* It is important to specify the second dimension of the array and the total number of columns is 50.
* The second parameter rows tell us about the total number of rows in the given two dimensional string array str.

Passing two dimensional String to a function:-

* To pass a two dimensional string to a function we have just write the name of the string array variable as the function argument.
* In the following example we have the name of 5 cities saved in an array cities of type char.
* We will be using the displayCities function to print the names.

Example:-

```
#include <stdio.h>
void displayCities (char [][50], int rows);

int main (void)
{
  char cities[][50]={
                      "chennai",
                      "New Delhi",
                      "Kolkata",
                      "Mumbai",
                      "Pune"
                    };
  int rows=5;
  displayCities (cities, rows);
  return 0;
}
void displayCities (char str[][50], int rows)
{
    int r, i;
    printf("Cities:\n");
    for(r=0; r<rows; r++)
    {
      i=0;
      while (str[r][i] != '\0')
      {
        printf("%c", str[r][i]);
        i++;
      }
      printF("\n");
}
```

Output:-

Cities:

Chennai

New Delhi

Kolkatta

Mumbai

Pune.