Nikhitha
EEE-A
250

STRINGS

Def: collection of characters or group of characters or set of characters or character array is known as string.

- In c language every string is terminated with null character ('\0').

Declaration of string:

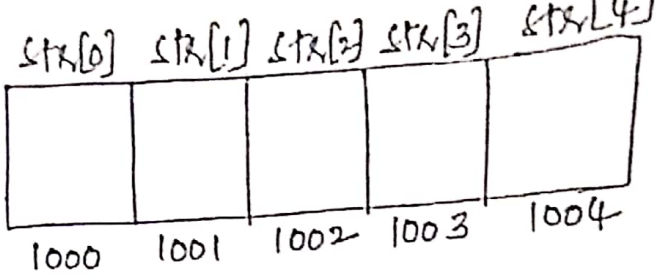
- Declaring a string is as simple as declaring a one dimensional array. Below is the basic syntax for declaring a string.

```
char str_name[size];
```

- In the above syntax str_name is any name given to the string variable and size represents the number of characters to be stored.
- To store a string of length 5, we need to specify 6 as the size (one extra size for null ('\0') character.
- Example for declaring a string is

```
char str[5];
```

- for the above character array str, the memory will be allocated as follows.

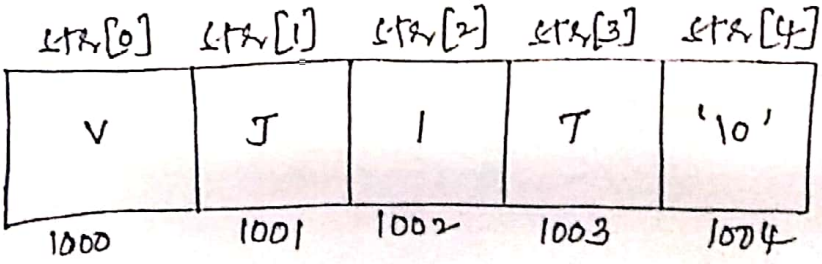


Initializing a string:

- A string can be initialized in the following different ways.

1. char str[5] = "VJIT";

Below is the memory representation of str



- Like we use subscripts & index to access the elements of an array, similarly subscripts are also used to access the elements of the character array.
- The subscript & index starts with zero
- All the elements & characters of character array are stored in successive memory locations.

2. char str[ ] = "VJIT";

- For the above statement the size is assigned by the compiler automatically based on the number of characters in a string.

3. char str[ ] = {'V','J','I','T','\0'};

In this example, we have explicitly added the null character. Here, the compiler will automatically calculate the size based on the number of characters initialized.

4. char str[5] = {'V','J','I','T','\0'};

Reading Strings:

- If we declare a string as

```
char str[5];
```

Then we can read str in 3 ways.

1. By using scanf() function:

```
scanf("%s", str);
```

- The main problem with scanf() function is that the function terminates as soon as it finds a blank space i.e when

Scanned by CamScanner

scanf() encounters a white space character, it terminates reading the string further.

- For example if we enter VIT, then str will contain VIT only.

2. By using gets() function:

- The next method of reading a string is by using gets() function.

- gets(str);

gets() is a simple function that overcomes the drawback of scanf() function.

3. By using getchar() function:

getchar() function is used to read a string character by character.

ex:

```
i=0;
ch = getchar();
while (ch != '*')
{
    str[i] = ch;
    i++;
    ch = getchar();
}
str[i] = '\0';
```


Scanned by CamScanner

Writing (&) displaying strings:

- We can display strings on screen in 3 ways.

1. By using printf() function:

- A string can be displayed using printf() function as follows:

```
printf("%s", str);
```

where str is character array name.

2. By using puts() function:

The next method of writing a string is by using puts() function. The string can be displayed by writing.

```
puts(str);
```

3. By using putchar() function:

We can display a string by using putchar() function as follows:

```
i=0;
while(str[i] != '\0')
{
    putchar(str[i])
    i++;
}
```


Scanned by CamScanner

String handling functions:

- The following are the string handling (or) string manipulation functions:

1. strlen():

This function is used to find length of the string.

Syntax:

```
n = strlen(string);
```

Ex: `n = strlen(str);`

where n is an integer variable which receives the value of the length of the string.

2. strcat():

This function is used to concatenate two strings.

Syntax:

```
strcat(str1, str2);
```

where str1 and str2 are character arrays.

- When strcat() function is executed, str2 is appended to str1.

- we must make sure that the size of str1 is large enough to accommodate the final string.
- strcat() function may also append a string constant to a string variable.

Ex: strcat (str1, "Hyd");

- C language permits nesting of strcat() functions.

Ex:

strcat (strcat (str1, str2), str3);

is allowed and concatenates all the 3 strings together. The resultant string is stored in str1.

3. strcmp() function:

- strcmp() function compares two strings and it will returns a value 0 (zero) if two strings are equal. If two strings are not equal, then it will returns the numeric difference between the first nonmatching characters.

Scanned by CamScanner

Syntax:

strcmp (string1, string2);

Ex:

strcmp (str1, str2);

- In the above syntax string1 and string2 may be string variables or string constants.

Ex:

strcmp (str1, str2);

strcmp (str1, "Hyd");

strcmp ("vjit", "Hyd");

- strcmp ("15eit", "15ece");

will returns a value of -9 which is the numeric difference between ASCII "i" and ASCII "R" i.e i minus R in ASCII code is -9.

4. strcpy():

- This function is used to copy one string into another string.

Syntax:

strcpy (string1, string2);

Scanned by CamScanner

Ex: strcpy (str1, str2);

will copy str2 string into str1 string.

- In the above example str2 may be a character array or string constant.

Ex: strcpy (str1, str2);

strcpy (str1, "VJIT");

5. strrev() function:

This function is used to reverse the given string.

Syntax: strrev (string);

Ex: strrev (str);

```
#include <stdio.h>
#include <string.h>
void main()
{
    char str[10];
    printf("Enter any string \n");
    gets(str);
    strrev(str);
    printf("\n reversed string = %s", str);
}
```


Scanned by CamScanner

Nikhitha
EEE-A
250

STRUCTURES

Def:

A Structure is a user defined data type, which is a collection of elements of different types.

- When we want to represent multiple values of different types as a single unit then we use structures.

Structure declaration & definition:

- A structure is declared using the keyword struct followed by a structure name.
- Syntax to define a structure is as follows.

```
struct structure-name (or) structure tag
{
    datatype member1;
    datatype member2;
      .
      .
      .
    datatype membern;
};
```


Scanned by CamScanner

Eg: 1

```
struct student
{
    int rno;
    char name[10];
    float fees;
};
```

Eg: 2.

```
struct employee
{
    int eid;
    char name[10];
    float sal;
};
```

- All the variables of the structure are declared within the structure.
- Each variable declared within a structure is called a member of the structure.

structure variable declaration:

- The structure declaration does not allocate any memory & consume storage space.

Scanned by CamScanner

- memory is allocated for the structure when we declare a variable of the structure.
- There are 2 ways in which we can declare a variable of structure. i.e

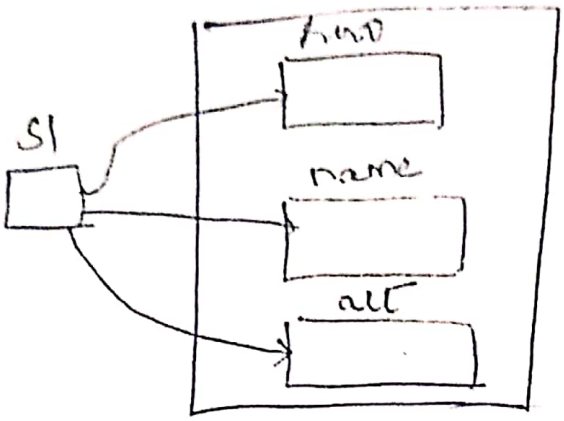
1. We can declare a variable to the structure at the time of structure declaration.

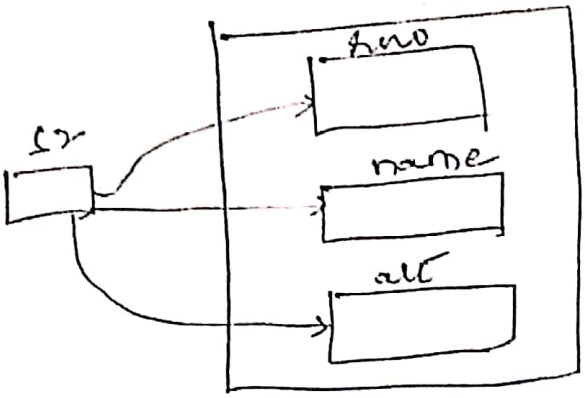
Ex:

```
struct student
{
    int rno;
    char name[10];
    float alt;
} s1, s2;
```

- In the above example s1 and s2 are variables of student type structure.
- When we declare variables of the structure, separate memory is allocated for each structure variable as shown below.

Scanned by CamScanner





2. we can declare variable to the structure by using the following syntax.

struct structure-name structure-var.name;

Ex:

struct student s1;

Initialization of structures:

- Initializing a structure means assigning some values to the members of the structures.

Scanned by CamScanner

- The syntax to initialize a structure variable is as follows:

```
struct structure_name
{
    datatype member1;
    datatype member2;
      :
    datatype membern;
} structure_variable = { constant1, constant2,
  constant3 .... };
```

(or)

```
struct structure_name
{
    datatype member1;
    datatype member2;
      :
    datatype membern;
};
struct structure_name structure_var
  = { constant1, constant2, constant3.... };
```


Scanned by CamScanner

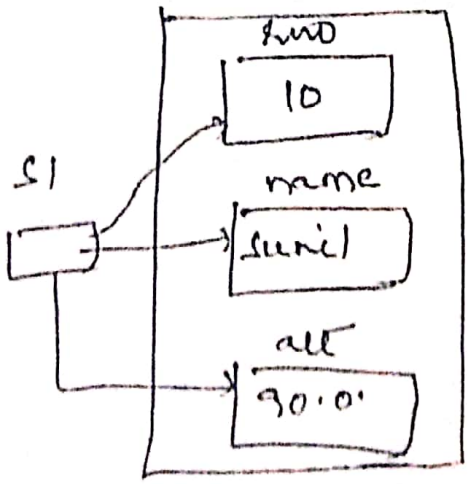
Ex:

```
struct student
{
    int rno;
    char name[10];
    float att;
} s1 = {10, "sunil", 90.0};
```

(OR)

```
struct student
{
    int rno;
    char name[10];
    float att;
};
struct student s1 = {10, "sunil", 90.0};
```

```
struct student
{
char name[10];
int rno;
float att;
};
struct student s = {"sunil",
                    10, 90.0};
```



Accessing the members of a structure:

- We can access structure member by using a . (dot) operator.
- The syntax to access members of a structure is as follows:

structure-variable.membername;

Ex:

s1.name
s1.sno
s1.all.

(OR)

```
struct student
{
    char name[10];
    int sno;
};
struct student s1;
s1.name = "Ravi";
s1.sno = 10;
```

The above is an example to assign value to the individual data members.

struct student
{
char name[10];
int sno;
};

struct student s1;
s1.name = ravi;
s1.sno = 50;

Scanned by CamScanner

- To read the values of structure members from keyboard we use the following code.

```
scanf("%s", s1.name);
scanf("%d", &s1.rno);
```

- To print & display values of structure members we use the following code.

```
printf("name = %s", s1.name);
printf("Rno = %d", s1.rno);
```

- Another way of accessing structure members by using → (arrow) operator.

Note:

- When we want to access structure members by using structure variable then we use . (dot) operator.
- When we want to access structure members by using structure pointer then we use → (arrow) operator.

Scanned by CamScanner

Nested Structures:

- A structure can be placed within another structure i.e a structure may contain another structure as its member.
- A structure that contain another structure as its member is called a nested structure.

Ex:

```
struct student
{
    char name[10];
    int rno;
    struct DOB
    {
       int day;
       int month;
       int year;
    }d;
}s;
```

- In the above example we are defining DOB structure as a member of student structure.
- In the above example student is outer structure and DOB is inner structure.

Scanned by CamScanner

- Another way of defining nested structure is as follows.

```
struct DOB
{
    int day;
    int month;
    int year;
};
struct student
{
    char name[10];
    int hno;
    struct DOB d;
};
```

- To access members of inner structure we use the following syntax.

outerstructure_var.innerstructure_var.membername;

Eg:

s.d.day

s.d.month

s.d.year

- To access outer structure members, we use the following code.

s.name , s.hno.

Scanned by CamScanner

Typedef :

- The typedef keyword enables the programmer to create a new data type name from an existing data type.
- By using typedef, no new data type is created, rather an alternate name is given to a known data type.
- The syntax of typedef is as follows:

typedef existingdatatype newdatatype;

- Ex:

```
typedef int INTEGER;
```

INTEGER is the new name of data type int.

- To declare a variable using the new name we write the following statement.

```
INTEGER V;
```

- The following is an example to define structure using typedef.

Scanned by CamScanner

```
typedef struct student
{
    char name[10];
    int rno;
}, stud;
```

In the above example stud is the synonym to the structure name student.
- By using stud we can declare a variable of student structure

Ex: stud s;

Enumerated data type:

- The enumerated data type is a user-defined type based on the standard integer type.
- An enumeration consists of a set of named integer constants. In other words, in an enumerated type, integer value is assigned to each identifier.
- To define enumerated data type, we use the keyword enum.

Scanned by CamScanner

- The syntax of creating an enumerated data type is as follows.

enum enumeration-name { identifier1, identifier2, ... identifiern };

Eg:

enum day { sun, mon, Tue, wed, Thur, fri, & sat };

- In the above syntax identifier1, identifier2, ... identifiern are also known as enumeration constants.
- If we do not assign any value to an identifier, the default value is assigned.
- The default value to the first identifier is 0. The rest of the undefined identifiers have a value 1 more than its previous one.
- for example, in sun, 0 is assigned, mon = 1, Tue = 2, wed = 3, Thur = 4, fri = 5, sat = 6.

Scanned by CamScanner

- If we want to assign values explicitly, then we can assign as follows:

```
enum day { sun, mon, Tue=5, wed,
    Thur, Fri, Sat };
```

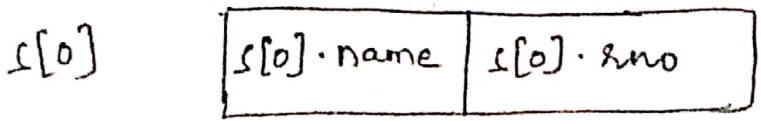
Array of structures:

- Declaring an array of structures is same as declaring an array of fundamental types.
- In an array of structures, each element of an array is of the structure type.

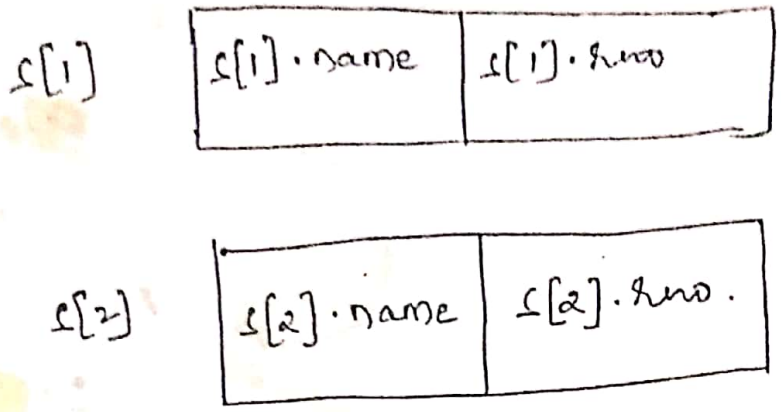
Eg:
```
struct student
{
    char name[10];
    int sno;
};
```

The following is the statement by which we can declare an array of structure student.

```
struct student s[3];
```



Scanned by CamScanner



- array of structures is nothing but collection of structures.

Array within structure:

- sometimes, array may be the member within structure, this is known as array within structure.
- using array inside the structure (as a member of the structure) is known as array within structure.

Ex:

```
struct student
{
    char name[10];
    int sno;
    int m[6];
};
```

- m[6] is now a member of structure student and to access this array we can we use . (dot) operator.

Scanned by CamScanner