Year : I

Semester : II

Subject Title : C PROGRAMMING

Sub Code : 18BCS23C

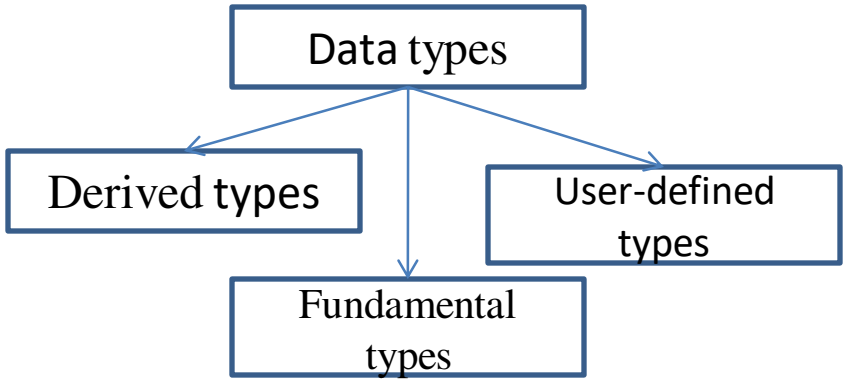
UNIT – III

CHAPTER VII

ARRAY

7.1 INTRODUCTION

- Arrays a kind of data structure that can store a fixed-size sequential collection of elements of the same type.
- Array can be used to represent not only simple lists of values but also tables of data in two, three or more dimensions
- Data structures



Derived types
- Array
- Functions
- Pointers

Fundamental types
- Integral types
- Float types
- Character types

User-defined type
- Structures
- Unions
- Enumerations
- Arrays and structures are referred to as structured data types because they can be used to represent data values that have a structure of some sort.

7.2 ONE-DIMENSIONAL ARRAY

- Creation and manipulation of the following data structures:

➢ Linked lists

➢ Stacks

➢ Queues

➢ Trees

Array types

- One-dimensional array
- Two-dimensional array
- multi-dimensional array

One dimensional array

- A list of items can be given one variable name using one subscript and such a variable is called a single-subscripted variable or a one-dimensional array
- one-dimensional array variable can be expressed as

For example

x[0],x[1],….x[n]

- The subscript can begin with number 0.

Declaration of one-dimensional array

Syntax

type variable-name[size];

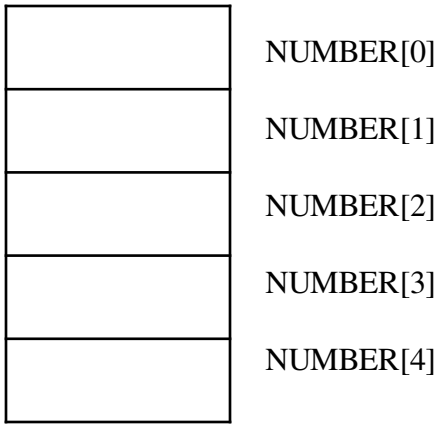
- The type specifies the type of element that will be contained in the array.
- Declares the number s an array to contain a maximum of 5 integer constants.

- Any reference to the arrays outside the declared limits would not necessarily cause an error. rather, it might result in unpredictable program results.
- The size should either numeric constant or symbolic constant.

Example

int number[5];

- The compiler reserves five storage locations



Initialization of one-dimensional array

- An array can be initialized at either compile time or run time.
- Compile time initialization

Syntax

type array-name[size]={list of values};

- The values in the list are separated by commas.

  int number[3]={0,0,0};
- Will declare the variable number as an array of size 3 and will assign zero to each element.
- If the number of values in the list is less than the number elements, then only that many elements will be initialized.
- The remaining elements will be set to zero automatically.
- The size may be omitted.

Example

float total[5]={0.0,15.57,-10};

int c[]={1,2,3,4};

- The c array contain 4 elements with initial values 1,2,3 and 4.

Character array

char arrayname[]={list of items};

Example

char name[]={'g', 'a', 'c', 'c', 'b', 'e'};

- Compile time intializzation may be partial. Ie.,the number of initailizers may be less than the declared size.
- The remaining elements are initialized to zero, if the array type is numeric and NULL if the type is char.

int a[5]={10,20};

- Will initialize the first two elements to 10 and 20 respectively and the remaining elements to zero.

Run time initialization

```
-------
-------
for (i=0;i<=100;i++)
{
if (i<50)
   sum[i]= 0;
else
    sum[i]=1;
}
-------
-------
```

- The first 50 elements are initialized to zero and remaining 50 elements are initialized to 1 at run time.

Example program

```
/*  program to find the percentage of marks */
#include <stdio.h>
#include <conio.h>
void main()
{
int rno,total,i,j,n,mark[5];
float percentage;
char name[25];
clrscr();
printf("Enter the no. of students\n");
scanf("%d",&n);
printf("Enter the student details one by
      one\n");
for (i=0;i<n;i++)
{
 printf("Enter student %d details\n",i+1);
printf("Enter the Rno :");
scanf("%d",&rno);
printf("Enter Name :");
scanf("%s",name);
printf("Enter five subject marks");
total=0;
for (j=0;j<5;j++)
{
scanf("%d",&mark[i]);
total=total+mark[i];
}
percentage=(float)total/5;
printf("RNO:",rno);
printf("\nName:",name);
prinf("Five subject mark\n);
for (j=0;j<5;j++)
      printf("\nsubject %d marks :
      %d",j+1,mark[j]);
printf("\nTotal : %d",total);
printf("\nPercentage : %f",percentage);
}
getch();
}
```

7.2.1 SORTING & SEARCHING

Sorting

- Sorting is the process of arranging in the list according to their vales, in ascending or descending order.
- A sorted list is called an ordered list.
- The sorting techniques are

i. Bubble sort

ii. Selection sort

iii. Insertion sort

iv. Shell sort

v. Merge sort

vi. quick sort

Seraching

- Searching is the process of finding the location of the specified element in a list.
- The specified element is called search key
- If the process of searching finds a match of the search key with the list element value then the search said to be successful; otherwise, it is unsuccessful.
- The most common search techniques are

i. Linear search

ii. Binary search

EXAMPLE

```
/* Bubble sort code */
#include <stdio.h>
int main()
        {
         int array[100], n, c, d, swap;
  printf("Enter number of elements\n");
         scanf("%d", &n);
  printf("Enter %d integers\n", n);
  for (c = 0; c < n; c++)
           scanf("%d", &array[c]);
  for (c = 0 ; c < n - 1; c++)
          {
           for (d = 0 ; d < n - c - 1; d++)
           {
            if (array[d] > array[d+1]) /* For decreasing
       order use '<' instead of '>' */
             {
              swap       = array[d];
              array[d]   = array[d+1];
              array[d+1] = swap;
             }
           }
          }
  printf("Sorted list in ascending order:\n");
  for (c = 0; c < n; c++)
            printf("%d\n", array[c]);
  return 0;
        }
```

```
/*selection sort*/
#include <stdio.h>
int main()
        {
         int array[100], n, c, d, position, t;
  printf("Enter number of elements\n");
         scanf("%d", &n);
  printf("Enter %d integers\n", n);
  for (c = 0; c < n; c++)
           scanf("%d", &array[c]);
  for (c = 0; c < (n - 1); c++) // finding minimum element (n-1) times
          {
           position = c;
   for (d = c + 1; d < n; d++)
           {
            if (array[position] > array[d])
              position = d;
           }
           if (position != c)
           {
            t = array[c];
            array[c] = array[position];
            array[position] = t;
           }
          }
  printf("Sorted list in ascending order:\n");
  for (c = 0; c < n; c++)
           printf("%d\n", array[c]);
  return 0;
        }
```

EXAMPLE

```
/*linear search*/
#include <stdio.h>
int main()
        {
         int array[100], search, c, n;
  printf("Enter number of elements in array\n");
         scanf("%d", &n);
  printf("Enter %d integer(s)\n", n);
  for (c = 0; c < n; c++)
           scanf("%d", &array[c]);
  printf("Enter a number to search\n");
         scanf("%d", &search);
  for (c = 0; c < n; c++)
         {
          if (array[c] == search)    /* If required element is found
        */
           {
            printf("%d is present at location
        %d.\n", search, c+1);
             break;
           }
         }
         if (c == n)
           printf("%d isn't present in the array.\n", search);
  return 0;
        }
```

```
/*binary serach*/
#include <stdio.h>
int main()
        {
         int c, first, last, middle, n, search, array[100];
  printf("Enter number of elements\n");
         scanf("%d", &n);
  printf("Enter %d integers\n", n);
  for (c = 0; c < n; c++)
           scanf("%d", &array[c]);
  printf("Enter value to find\n");
         scanf("%d", &search);
  first = 0;
         last = n - 1;
         middle = (first+last)/2;
  while (first <= last) {
           if (array[middle] < search)
            first = middle + 1;
           else if (array[middle] == search) {
            printf("%d found at location %d.\n", search, middle+1);
            break;
           }
           else
            last = middle - 1;
   middle = (first + last)/2;
         }
         if (first > last)
           printf("Not found! %d isn't present in the list.\n", search);
  return 0;
        }
```

7.3 TWO-DIMENSIONAL ARRAYS

Two-dimensional array

-Rows

-Columns

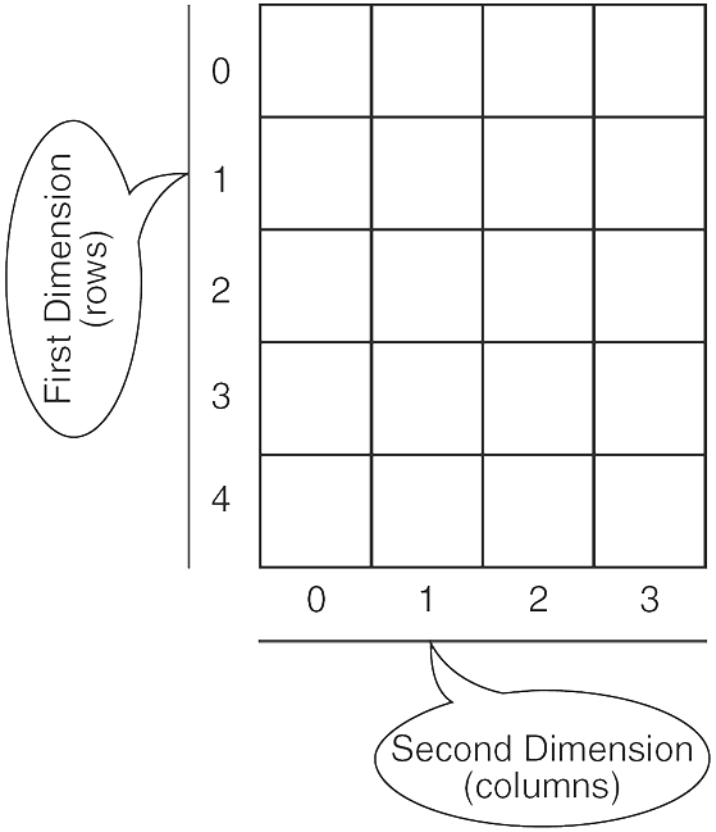
- A list of items can be given one variable name using two subscripts and such a variable is two-dimensional array.

Declaration of two-dimensional array

Syntax

Type array_name[row-size][column-size];

- Logically it may be viewed as a two-dimensional collection of data, three rows and five columns, each location is of type int.



INITIALIZING TWO-DIMENSIONAL ARRAY

Syntax

type array-name[row-size][colum-size]={list of values};

Example

1. ***int a[2][2]={1,2,3,4};***

- First row initialized to 1,2 and second row is 3,4.

2. ***int b[2][2]={{1,1},{2,2}};***

- The initialization is done row by row.
- First row elements initialized to 1 and second row elements initialized to 2.

3. ***int a[2][3]={***
{0,0,0},
{1,1,1}
};

- The array is initialized in the form of matrix.

4. ***int a[][3]={***
{0,0,0},
{1,1,1}
};

is permitted.

5. If the values are missing in an initializer, they are automatically set to zero.

in a[2][3]={{1,1},{2}};

- Will initialize the first elements of the first row to 1then the first element of the second row to two and all other elements to zer0.

6. ***int a[2][3]={{0},{0},{0}}***

- All the elements are set to zero.

EXAMPLE PROGRAM

```
#include <stdio.h>
#include <conio.h>
void main()
{
int a[5][5],b[5][5],c[5][5],i,j,n,m,k,l;
loop:clrscr();
printf(“\n Enter the size of the first matrix”);
scanf(“%d%d”,&n,&m);
printf(“\n Enter the size of the second matrix”);
scanf(“%d%d”,&k,&l);
If ((k!=n) && (m!=l))
      {
        printf(“Matrix addition cannot perform\n”);
          printf(“Enter the size of first matrix and
      second matrix are equal\n);
       goto loop;
       }
printf("\nEnter values for matrix A:\n");
for(i=0;i<n;i++)
for(j=0;j<m;j++)
      scanf("%d",&a[i][j]);

printf(“\nEnter values for matrix b:\n");
for(i=0;i<k;i++)
for(j=0;j<l;j++)
      scanf("%d",&b[i][j]);

for(i=0;i<n;i++)
for(j=0;j<m;j++)
      c[i][j]=a[i][j]+b[i][j];
printf("\nMatrix c elements are:\n");
for(i=0;i<n;i++)
{
      for(j=0;j<m;j++)
            printf(“%d\t",c[i][j]);
      printf("\n");
}
getch();
}
```

7.4 MULTI-DIMENSIONAL ARRAY

- C allows three or more dimensions.
- The exact limit is determined by the compiler.

Syntax

type array-name[s1][s2]….[sn];

- Where si is the size of the ith dimension.

Example

int a[2][3][4];

- 'a' is multi-dimensional array which contains 24 integer type elements.
- Three dimensional array can be represented as a series of two-dimensional arrays in c.
- ANSI C does not specify any limit for array dimensions.
- Most compilers permit only 7 to 10 dimensions.
- Some compiler allows more.

7.5 DYNAMIC ARRAY

- The process of allocating memory at compile time is called static memory allocation and the array that receive static memory allocation are called static arrays.
- The process of allocating memory at runtime is called dynamic memory allocation and the array that receive dynamic memory allocation are called dynamic arrays.
- Dynamic arrays are created using pointer variables and memory management functions malloc, calloc and realloc.
- These functions are included in stdlib.h.
- The concept of dynamic arrays is used in creating and manipulating data structures such as linked lists, stacks and queues.

EXAMPLE: BUBBLE SORT

```
/* Bubble sort code  and find the median */
#include <stdio.h>
void main()
{    int n,i,j;
   float a[100],median,temp;
printf("Enter number of elements\n");
       scanf("%d",&n);     printf("Enter %d
       integers\n",n);
for (i=0;i<n;i++)
  scanf("%f",&a[i]);
 for (i=0;i<n;i++)
  {       for(j=i+1;j<n;j++)
    {          if (a[i] > a[j])
                {
                 temp=a[i];
                 a[i]=a[j];
                 a[j]=temp;
                 }
else
                    continue;
       }
  }

/* median*/
 if (n%2 ==0)
         median= (a[n/2]+a[n/2+1])/2.0;
  else
         median=a[n/2];
 printf("Sorted list in ascending order:\n");
 for (i=0;i<n;i++)
     printf("%f\n",a[i]);
   printf("Median= %f",median);
  getch();
}
```

OUTPUT

OUTPUT:

```
Enter number of elements
5
Enter 5 integers
5
3
2
4
7
Sorted list in ascending order:
2.000000
3.000000
4.000000
5.000000
7.000000
Median= 4.000000
```

UNIT – III

CHAPTER VIII

CHARACTER ARRAYS & STRINGS

8.1INTRODUCTION

- A string is a sequence of characters that is treated as a single data item.
- The common operations performed on character strings are

i. Reading and writing strings
ii. Combining strings together
iii. Copying one string to another
iv. Comparing string for equality
v. Extracting a portion of a string

Declaring and initializing string variables

Syntax

char string_name[size];

- The size determines the number of characters in the string_name.

Example

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 array can be initialized when they are declared.
- Character array can be initialized in either of the following

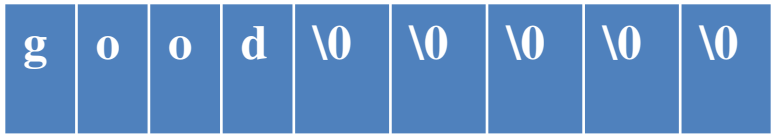
char city[11]="coimbatore";

char city[11]={'c', 'o', 'i', 'm', 'b', 'a', 't', 'o', 'r', 'e'};

char city[]={'c', 'o', 'i', 'm', 'b', 'a', 't', 'o', 'r', 'e'};

- We can declare the size much larger than the string size in the initializer.

char str[9]="good";



char str[3]="good"; ← illegal

char str[5];

str="good";

← illegal

The string is a variable-length structure and is stored in a fixed-length array. Therefore, the last element of the array need not represent the end of the string. The null character serves as the "end-of-string".

Reading string from the terminal

(i) Using scanf() function

chat str[25];

scanf("%s",str);

- The ampersand (&) is not required before the variable name str.
- Scanf() function terminates the input on the first white space it finds.
- A white space includes blanks, tabs, carriage returns, form feeds, and new lines.
- If input is Coimbatore – 18 then only Coimbatore will be read into the array str, since the blank space after the Coimbatore will terminates the reading of string.
- The scanf() function automatically terminates the string that is read with a null character and therefore the character array should be large enough to hold the input string plus a null character.
- The unused locations are filled with garbage.
- We can specify the field width using the form %ws in the scanf() statement for reading a specified no. of characters from the input string.

scanf("%ws",str);

- 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 w is less than the number of characters typed in the string. The excess characters will be truncated and left unused.

Example

char name[10];

scnaf("%5s",name);

- If the input string is RAM will be stored as

R	A	M	\0	?	?	?	?	?	?
0	1	2	3	4	5	6	7	8	9

- If the input string is KRISH will be stored as

K	R	I	S	H	\0	?	?	?	?
0	1	2	3	4	5	6	7	8	9

Reading a line of text

char line[100];

scnaf("%[^\n]",line);

printf("%s",line);

- Will read a line of input from the keyboard and display the same on the screen.

(ii) Using getchar() and gets()

a) getchar()

char ch;

ch=getchar();

- Reads a character from the user and assign to the variable 'ch'.
- We can use this function to repeatedly to read successive single characters from the input and place them into a character array.
- An entire line of text can be read and stored in an array..

This reading is terminated when the newline character is entered.

Null character is inserted at the end of the string.

b) gets()

- It reads a characters into 'str' from the keyboard until a new line character is encountered and then appends a null character to the string.
- Unlike scanf() function, it does not skip white spaces.

Example

char str[100];

gets(str);

printf("%s",str);

(or)

char str[100];

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

- C does not provide operators that work on strings directly.
- We cannot assign one array to another directly.

Example

St1="abc";

Str2=str1;→ illegal

- Example program using scanf() and getchar() function to read a string.

Writing strings to screen

(i) Using printf() function

- The printf() function with %s format to print strings to the string.
- The format %s can be used to display an array of characters that is terminated by null character.
- We can specify the precision with which the array is displayed
- %w.ps indicates that the first four characters are to be printed in a field width of 10 columns.
- %-ws indicates that the string will be printed right-justified.

(ii) putchar() and puts() function

(a) puctchar()

- The function putchar() requires on parameter.
- Write a character to the screen.
- We can use this function repeatedly to output a string of characters stored in an array using a loop.

Example

```
char ch='a';
putchar(ch);
printf("%c",ch);
```

Example 2

```
char name[8]=“chennai”;
for (i=0;i<7;i++)
    putchar(name[i]);
putchar(‘\n’);
```

(b) puts()

- Printing the string values.

Example

puts(str);

- Where str is a string variable containing a string value.
- This prints the value of the string variable str and then moves the cursor to the beginning of the next line on the screen.

Example using scanf() and printf()

```
#include <stdio.h>
#include <conio.h>
void main( )
 {
  char str[100];
  int i;
   printf( "Enter a string:");
  scanf("%s ", str, &i);
  printf( “\n String:  %s ", str);
 getch();
}
```

Example using gets() and puts() function

```
#include <stdio.h>
#include <conio.h>
void main( )
 {
 char str[100];
 printf( “Enter a string :");
 gets( str );
printf( “\n String: ");
 puts( str );
getch();
}
```

example using getchar() and putchar() function

```
#include <stdio.h>
#include < conio.h>
void main( )
 {
    char ch;
    printf( “Enter a String :");
    c = getchar( );
    printf( “\n String: ");
    putchar(c);
    getch();
}
```

8.3 ARITHMETIC OPERATIONS ON CHARACTERS

- whenever a character constant or character variable is used in an expression , it is automatically converted into an integer value by the system.
- The integer value depends on the local character set of the system.

Example

x='a';

printf("%d",x);

Output

97

- It is also possible to perform arithmetic operations on the character constants and variables.

Example

X='z'-1;

Printf("%d",x);

Output

121

Ie. The ascii value of z is 122.

We may use character constants in relational expressions.

Example

Ch>='A' && ch <='Z'

We can convert a character digit to its equivalent integer value using relationship.

X=c-'0';

Where x is defined as an integer variable and c contains the character digit.

Example

```
c='7';
x=c-'0';
printf("%d",x)
```

Output

7

where

X = ascii value of '7'- ascii value of '0';

= 55-48

=7

- 'c' library supports a function that converts the string of digits into their integer values.

Syntax

variable=atoi(string);

- Where variable is a integer variable and string is a character array containing a string of digits.

Example

```
n="12345";
y=atoi(n);
printf("y= %d",y);
```

Output

y= 12345

8.4 STRING HANDLING FUNCTIONS

- Some of the string handling functions in c.
- string.h

Function	Action
strcpy(s1,s2)	copies a **string** to another
strcat(s1,s2)	concatenates(joins) two strings
strcmp(s1,s2)	compares two strings
strlwr(s1)	converts **string** to lowercase
strupr(s1)	converts string to uppercase
strchr(s1, ch);	Returns a pointer to the first occurrence of character ch in string s1.
strstr(s1, s2);	Returns a pointer to the first occurrence of string s2 in string s1.
strlen(s1)	Returns the no. of character in string s1.

strcat() function

strcat(s1,s2);

- concatenates(joins) two strings.
- s1 and s2 are character arrays.
- When the function strcat() is executed s2 is appended to s1.
- It does so by removing of null character at the end of s1 and placing s2 from there.
- s2 remains unchanged.

Example

s1= 

s2= 

strcat(s1,s2);

S1= 

s2= 

strcmp() function

- strcmp() function compares two strings identified by the arguments.
- If they are equal then returns 0.
- If s1 is greater than s2 then returns +ve value.
- Otherwise, returns –ve value.

Syntax

strcmp(s1,s1)

strcpy() function

- The function copies the contents of s2 to s1.
- It is like assignment.

Syntax

strcpy(s1,s1);

- Assigns the contents of s2 to s1.
- s2 may be string array variable or a string constants.

strlen() function

- This function counts and returns the no. of characters in a string.

Syntax

n= strlen(s1);

- Where n is the integer variable, which receives the value of the length of the string.
- s1 may be a string constant.
- The counting ends at the null character.

EXAMPLE PROGRAM :1

```
#include <stdio.h>
#include <string.h>
 int main ()
 {
char str1[15];
 char str2[15];
 int ret;
strcpy(str1, "abcdef");
strcpy(str2, "abcdef");
ret = strcmp(str1, str2);
 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);
 }
```

EXAMPLE PROGRAM :2

```
#include <stdio.h>
 #include <string.h>
 int main ()
 {
   char str1[12] = "Hello";
   char str2[12] = "World";
   char str3[12];
   int len ;
   /* copy str1 into str3 */
   strcpy(str3, str1);
   printf("strcpy( str3, str1) : %s\n", str3 );
   /* concatenates str1 and str2 */
   strcat( str1, str2);
   printf("strcat( str1, str2): %s\n", str1 )
/* total lenghth of str1 after concatenation */
   len = strlen(str1);
   printf("strlen(str1) : %d\n", len );
   return 0;
}
```

Output

```
strcpy( str3, str1) : Hello
 strcat( str1, str2): HelloWorld
 strlen(str1) : 10
```

strncpy() function

- The function copies only the left-most n characters of the source string to the target string variable.

Syntax

strncpy(s1,s2,n);

- This statement copies the first n characters of s2 to s1.

strstr() function

- It can be used to locate a sub-string in a string.

Syntax

strstr(s1,s2);

- The function strstr searches the string s1 to see whether the string is contained in s1.
- If yes, the function returns the position of the first occurrence of the sub-***string.***
- Otherwise, it returns null pointer

strncmp() function

- A this function compares the left-most n characters of s1 to s2 and returns,

(i) O if they are equal

(ii) -ve no., if s1 substring is less than s2; and

(iii) +ve no, otherwise.

Syntax

strncmp(s1,s2,n);

strncat() function

This function concatenates only the left-most n characters of the source string to the target string variable.

Syntax

strncat(s1,s2,n);

```
#include <stdio.h>
 #include <string.h>
 int main ()
 {
char s[40];
char d[12];
 strcpy(s, "This is strcpy example");
strncpy(d, s, 10);
printf(“Destination string : %s\n", d);
 return(0);
 }
```

```
#include <stdio.h>
#include <string.h>
 int main ()
{
 char s1[15];
 char s2[15];
int ret;
strcpy(s1, “GAC CBE-18");
strcpy(s2, “gac cbe-18");
 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);
 }
```

```
#include <stdio.h>
#include <string.h>
 int main ()
 {
 char s[50],
d[50];
strcpy(s, "This is source string");
 strcpy(d, "This is destination string");
strncat(d, s, 15);
printf("Final destination string : |%s|", d);
return(0);
}
```

```
#include <stdio.h>
 #include <string.h>
 int main ()
 {
char a[20] = “GAC CBE -18";
char b[10] = “CBE";
char *ret;
ret = strstr(a, b);
printf("The substring is: %s\n", ret);
 return(0);
 }
```

8.5 TABLE OF STRING

- Two-dimensinal character array.

Syntax

char array-name[size1][size2];

Example

char name[10][20];

- The name array can stored a ten names with 20 characters length.

Example program

```
/* C Program to sort a list of strings  in
     ascending order */
#include <stdio.h>
#include <string.h>
int main ()
{
  char name[20][100];
char temp[20];
int i, j;
printf("Enter the no. of strings");
scanf("%d",&n)
printf("Enter the name one by one\n");
for(i=0;i<n;i++)
{
     scanf("%s",name[i]);
}
for (i = 0; i < n-1; i++)
 {
     for (j = i+1; j < n; j++)
      {
            if (name[i] > name[j])
             {

            strcpy(temp,name[i]);
            strcpy(name[i],name[j]);
            strcpy(name[j],temp);
               }
        }
     }
     printf("The sorted name list \n");
 for(i=0;i<n;i++)
     printf("%s\n",name[i]);
     return 0;
}
```

References:

- E. Balagurusamy, ”Programming in ANSI C”, Seventh Edition, McGraw Hill Education India Private Ltd, 2017.
- https://www.tutorialspoint.com/c_standard_library/c_function