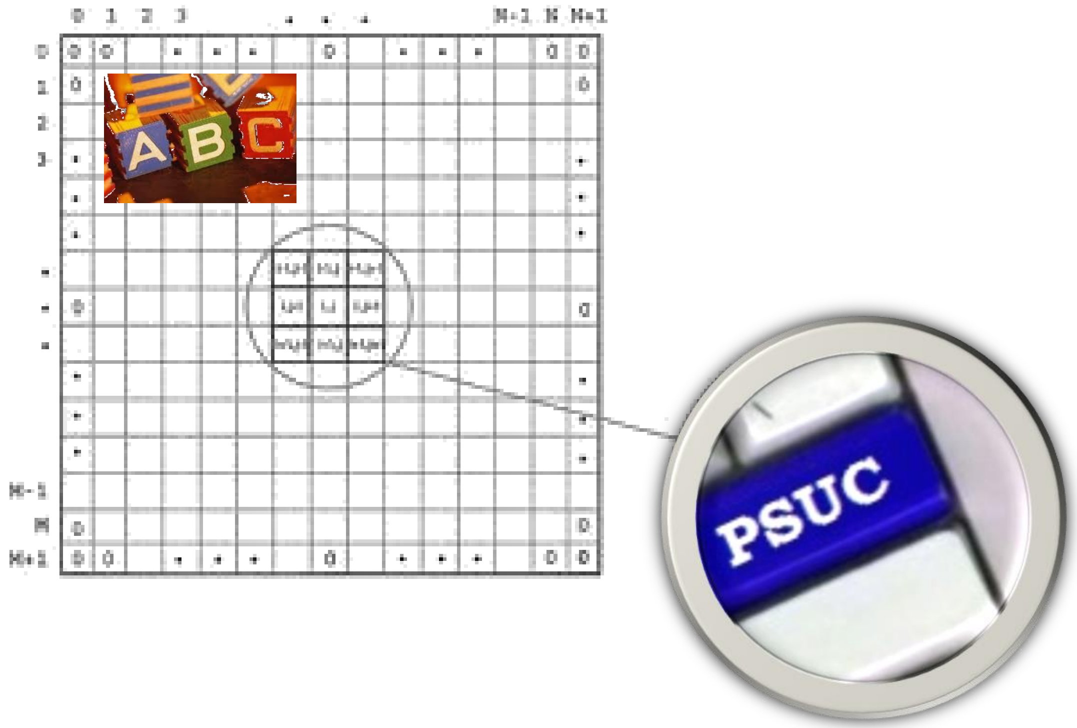


CHARACTER ARRAYS- STRINGS

Objectives

To learn and appreciate the following concepts

- **Strings definition, declaration, initialization**
- **Reading Strings**
- **String Handling Functions**
- **Programs using strings**
- **Array of Strings**
- **Operations on array of strings**

Session outcome

At the end of session student will be able to

- **Declare and initialize strings and array of strings**
- **Write programs using strings**

Strings

Definition

- A string is an array of characters.
- Any group of characters (except double quote sign) defined between double quotation marks is a **constant string**.
- Character strings are often used to build meaningful and readable programs.

The common operations performed on strings are

✓**Reading and writing strings**

✓**Combining strings together**

✓**Copying one string to another**

✓**Comparing strings to another**

✓**Extracting a portion of a string (substring)** ..etc.

Strings

Declaration and initialization

char string_name[size];

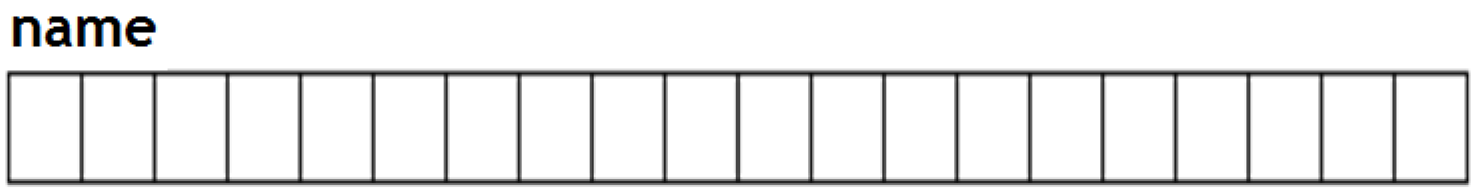
The size determines the number of characters in the string_name.

For example, consider the following array:

char name [20];

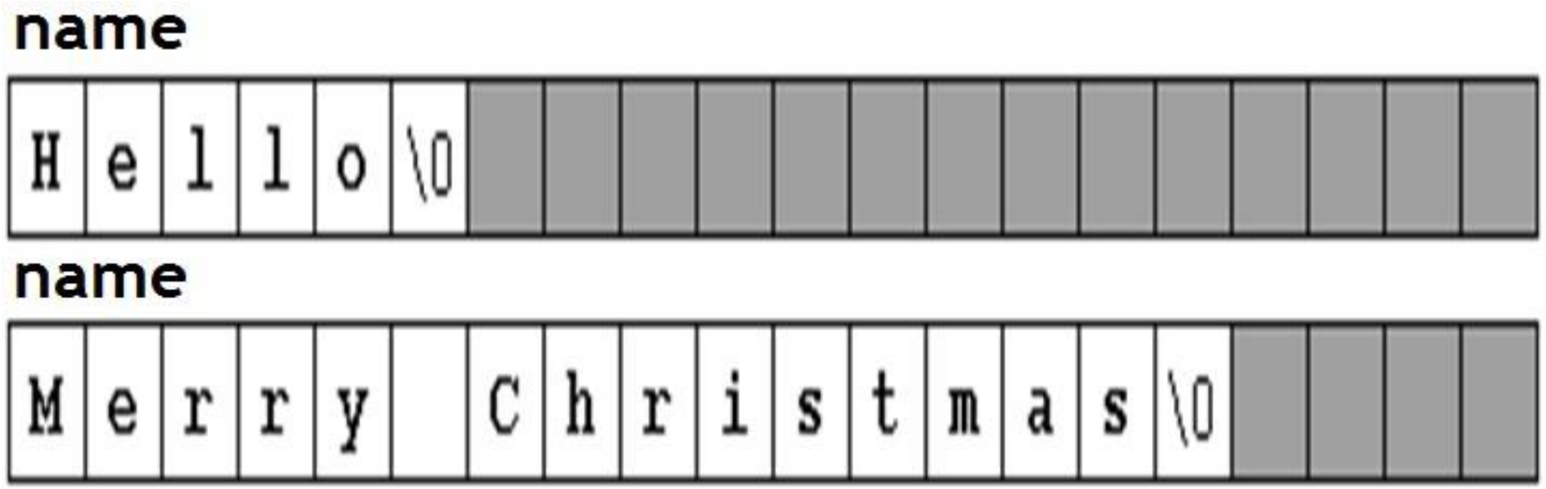
is an array that can store up to 20 elements of type char.

It can be represented as:



Strings

✓The character sequences "**Hello**" and "**Merry Christmas**" represented in an array name respectively are shown as follows :



Initialization of null-terminated character sequences

- **array of characters** or **strings** are ordinary arrays that follow the same rules of arrays.

For example

To initialize an array of characters with some predetermined sequence of characters, one can initialize like any other array:

```
char myWord[ ] = { 'H', 'e', 'l', 'l', 'o', '\0' };
```

Initialization of null-terminated character sequences

- Arrays of character elements have additional methods to initialize their values: **using string literals**
- **“Manipal ”** is a constant string literal.

  For example,

  char result[14] =“Manipal”;

- **Double quoted** (") strings are literal constants whose type is in fact a **null-terminated array of characters**.

  So string literals enclosed between double quotes always have a null character ('**\0**') automatically appended at the end.

Initialization

```
char myWord [ ] = { 'H', 'e', 'l', 'l', 'o', '\0' };

char myWord [ ] = "Hello";
```

- In both cases the array of characters myword is declared with a size of 6 elements of type char:

✓The 5 characters that compose the word "**Hello**" plus a final null character ('**\0**') which specifies the end of the sequence and that,

✓In the second case, when using double quotes (") null character ('**\0**') is appended automatically.

Example

```
#include <stdio.h>

int main() {

    char question[ ] = "Please, enter your first name: ";

    char greeting[ ] = "Hello, ";

    char yourname [ 80];

    scanf(“%s”,yourname); //format specifier: %s

    printf(“%s, %s\n”,greeting , yourname );

 return 0;

}
```

Example

```
#include <stdio.h>
int main() {
const int MAX = 80;//max characters in string
char str[MAX];         //string variable str
printf( “Enter a string: \n”);
scanf(“%s”,str);     //put string in str
printf(“%s”,str);    //display string from str
return 0;
}
```

Reading Embedded Blanks

To read everything that you enter from the keyboard until the ENTER key is pressed (including space).

Syntax:

```
gets(stringname) ;
```

To write/display that you entered.

Syntax:

```
puts(stringname) ;
```

Example

```
#include <stdio.h>
int main() {
 const int MAX = 80;     //max characters in string
 char  str[MAX];         //string variable str
 printf("\nEnter a string: ");
 gets(str);              //put string in str
 printf(" the string is \n");
 puts(str);
 return 0;
}
```

Count the number of characters in a string

```
#include <stdio.h>
int main() {
const int Max = 100;
char sent[Max];
int i=0, count=0;

printf("enter sentence \n“);
gets(sent);

puts(sent);
```

```
while(sent[i]!='\0') {
   count++;
   i++;
}
printf(“No of chars = %d“, count);
return 0;
}
```

Count the number of words in a sentence

```
#include <stdio.h>
int main() {
const int MAX = 100;
char sent[MAX];
int i=0,count=1;

printf("enter sentence \n“);
gets(sent);
printf("\n“);
while(sent[i]!='\0')
{
   if(sent[i]==' '&& sent[i+1]!=' ')
    count++;
    i++;
}
printf(“No.of words = %d“, count);
return 0;
}
```

Reading multiple lines: Example

```
#include <stdio.h>
int main() {
const int MAX = 2000; //max characters in string
char str[MAX]; //string variable str
printf("\nEnter a string:\n“);
scanf("%[^#]",str); //read characters to str until a # character is encountered
printf("You entered:\n“);
printf(“%s”,str);
return 0;
}
```

The function will continue to accept characters until termination key (#) is pressed.

Library functions: String Handling functions (built-in)

- Used to manipulate a given string.
- These functions are part of string.h header file.
  - strlen ()
    - ✓ gives the length of the string. E.g. strlen(string)
  - strcpy ()
    - ✓ copies one string to other. E.g. strcpy(Dstr1, Sstr2)
  - strcmp ()
    - ✓ compares the two strings. E.g. strcmp(str1, str2)
  - strcat ()
    - ✓ Concatinate the two strings. E.g. strcat(str1, str2)

Library function: strlen()

- String length can be obtained by using the following function

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

- This function counts and returns the number of characters in a string, where n is an integer variable which receives the value of the length of the string.

- The argument may be a string constant.

  Eg: `printf(“%d”,strlen(“Manipal”));` prints out 7.

Copies a string using a `for` loop

```
#include <stdio.h>
#include<string.h>
int main() {
 char str1[ ] = “Manipal Institute of Technology”;
 const int MAX = 80;          //size of str2 buffer
 char str2[MAX];              //empty string
 for(int j=0 ; j<strlen(str1); j++) //copy strlen characters
       str2[j] = str1[j];          // from str1 to str2
 str2[j] = ‘\0’;              //insert NULL at end
 printf(“%s\n”,str);          //display str2
 return 0;
}
```

Library function: strcpy()

Copying a String the EASY WAY using

strcpy(destination, source)

- The strcpy function works almost like a string assignment operator and assigns the contents of source to destination.

✓destination may be a character array variable or a string constant.

e.g., `strcpy(city, ”DELHI”);`

will assign the string “DELHI” to the string variable city.

✓Similarly, the statement **strcpy(city1, city2);**

will assign the contents of the string variable city2 to the string variable city1.

The size of the array city1 should be large enough to receive the contents of city2.

strcpy(): Example

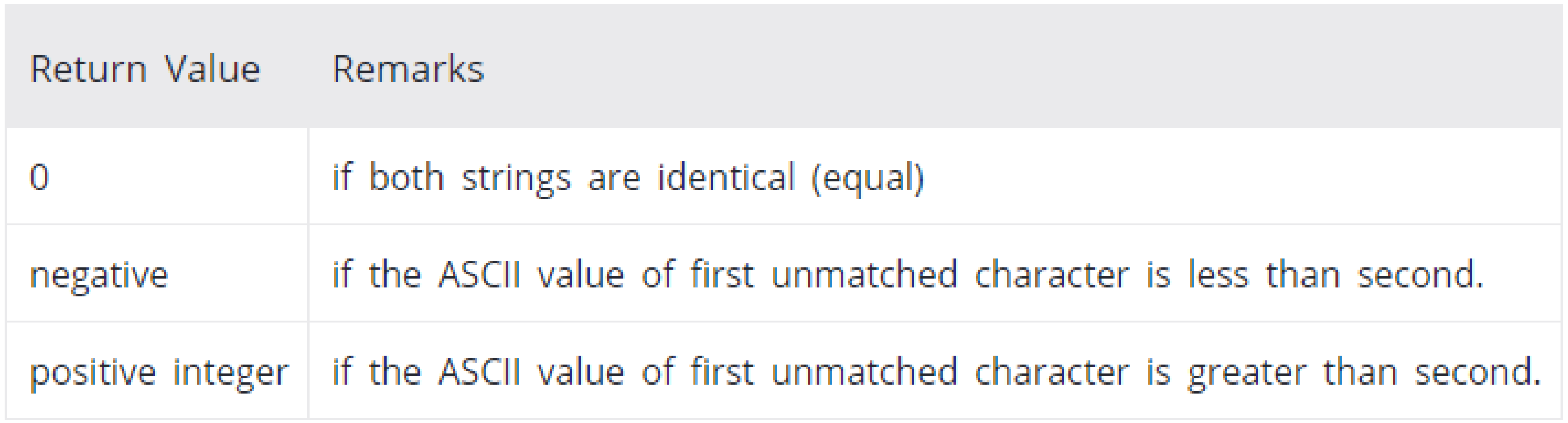
```
#include <stdio.h>

int main(){

char str1[ ] = "Tiger, tiger, burning bright\n"

          "in the forests of the night";

const int MAX = 80;  //size of str2 buffer

 char str2[MAX];  //empty string

strcpy(str2, str1);  //copy str1 to str2

printf("%s",str2);  //display str2

 }
```

Library function: strcmp()

- The **strcmp function** compares two strings identified by the arguments and has a value 0 if they are equal.

  `strcmp(string1, string2);` string1 and string2 may be string variables or string constants.

- If they are not **equal**, it returns the numeric values -1 or 1.



Return Value	Remarks
0	if both strings are identical (equal)
negative	if the ASCII value of first unmatched character is less than second.
positive integer	if the ASCII value of first unmatched character is greater than second.

That means, if the value is **negative**, string1 is alphabetically above string2.

Library function: strcat()

The **strcat function** joins two strings together.

It takes the following form:

```
strcat(string1, string2);
```

string1 and string2 are character arrays.

- ✓ When the function **strcat** is excuted, 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.

Concatenation of 2 strings

```
#include <stdio.h>
#include <string.h>
int main(){
   char s1[40], s2[50];
   printf("\nEnter the first string“);
   gets(s1);
   printf("\nEnter the second string“);
   gets(s2);
   strcat(s1, s2);
   printf("\nConcatenated string is“);
   printf(“%s”,s1);
   return 0;   }
```

Strings

Declaration and initialization

```
char string_name[size];
```

The size determines the number of characters in the string_name.

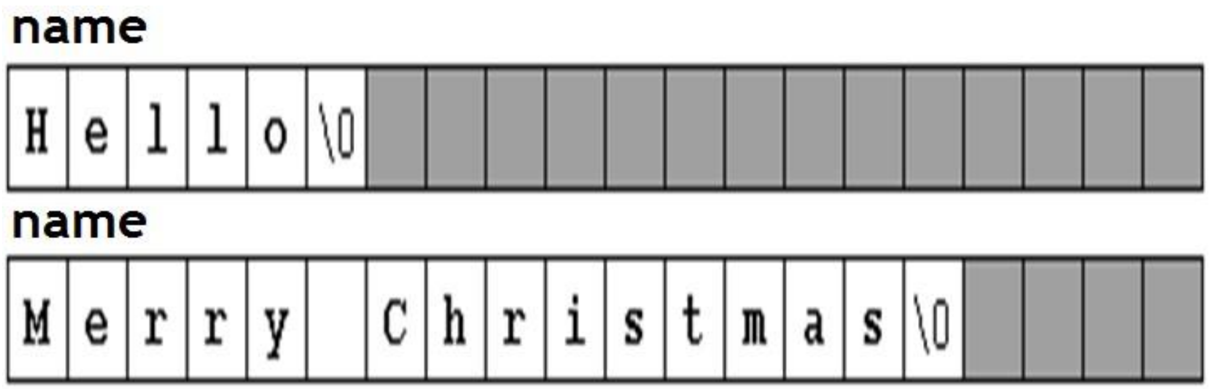
For example, consider the following array:

```
char name [20];
```

is an array that can store up to 20 elements of type char.

It can be declared, initialized & represented as:

```
char name[ ] = { 'H', 'e', 'l', 'l', 'o', '\0' };

char name[ ] =“Hello”;

char name[ ] =“Merry Christmas”;
```



Library functions: String Handling functions (built-in)

- Used to manipulate a given string.
- These functions are part of string.h header file.
  - strlen ()
    - ✓ gives the length of the string. E.g. strlen(string)
  - strcpy ()
    - ✓ copies one string to other. E.g. strcpy(Dstr1, Sstr2)
  - strcmp ()
    - ✓ compares the two strings. E.g. strcmp(str1, str2)
  - strcat ()
    - ✓ Concatinate the two strings. E.g. strcat(str1, str2)

Check whether a string is Palindrome or not

```
int main(){
char str[30];
int i, j, n, flag=1;
printf("\nEnter the string:");
gets(str);
//find the string length
for(i=0;str[i]!='\0';i++);
 n=i; //n=strlen(str);
```

```
for(i=0;i<n/2;i++){
   if(str[i]!=str[n-i-1])
     { flag=0;
       break; }
}
if(flag==1)
    printf("\nIts a  Palindrome");
else
    printf("\nNot a Palindrome");
return 0;
}
```

Reversing a string

```
int main()
{
char str[70];
char temp;
int i, n=0;
printf("\nEnter the string:");
gets(str);
for(i=0;str[i]!='\0';i++)
   n++;
```

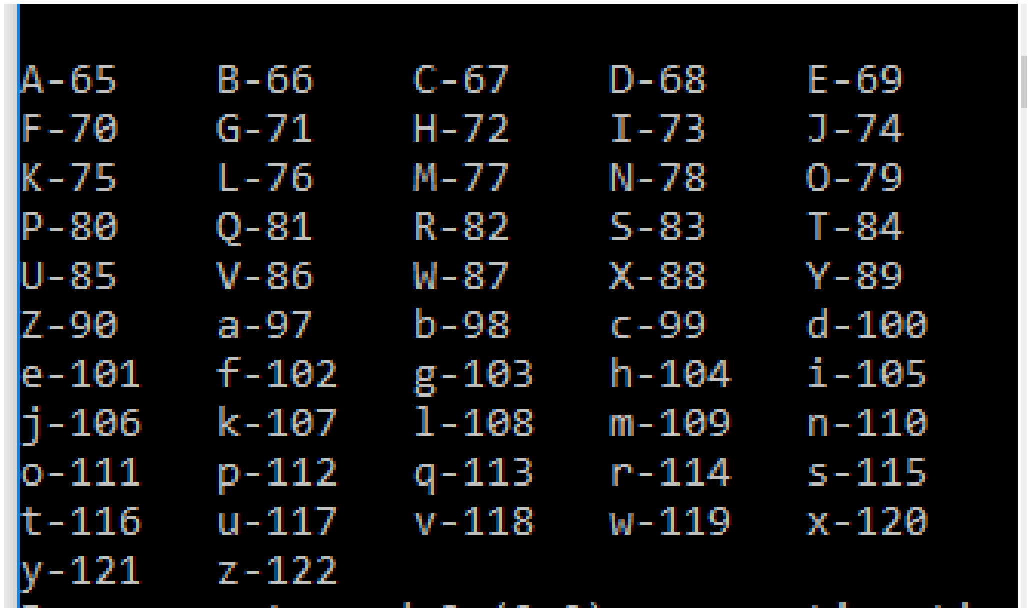
```
for(i=0;i<n/2;i++)
{
     temp=str[i];
     str[i]=str[n-i-1];
     str[n-i-1]=temp;
}
printf("\nReversed string is:");
puts(str);
return 0;
}
```

Password reading problem

```
int main() {
 char pw[10],un[10], ch;
 int i;
 printf("Enter User name: ");
 gets(un);
 printf("Enter password <6 char>:");
 for(i=0;i<6;i++) {
  pw[i] = getch();
  printf("*");
 }
/*print the entered password*/
 printf("\nYour password is : ");
 puts(pw);
return 0;
}
```

Print an alphabet in decimal [ASCII] & character form

```
int main()
{
char c;

printf("\n");
for(c=65;c<=122;c++)
{
 if(c>90 && c<97)
    continue;
printf("%c", c);
printf("-");
printf("%d",c);
printf("\t");
}
return 0;
}
```



```
A-65    B-66    C-67    D-68    E-69
F-70    G-71    H-72    I-73    J-74
K-75    L-76    M-77    N-78    O-79
P-80    Q-81    R-82    S-83    T-84
U-85    V-86    W-87    X-88    Y-89
Z-90    a-97    b-98    c-99    d-100
e-101   f-102   g-103   h-104   i-105
j-106   k-107   l-108   m-109   n-110
o-111   p-112   q-113   r-114   s-115
t-116   u-117   v-118   w-119   x-120
y-121   z-122
```

Change all lower case letters into uppercase in a sentence

```
int main()
{
char string[30];
int i,n=0;
printf("\nEnter the string");
gets(string);

for(i=0;string[i]!='\0';i++)
   n++;
```

```
for(i=0;i<n;i++)
{
  if(string[i]>=97 && string[i]<=122)
  string[i]=string[i]-32;
}

puts(string);
return 0;
}
```

Sorting n names in alphabetical order

```
int main(){
char string[30][30],temp[30];
int no, i, j;
printf(“Enter the no of strings:“);
scanf(“%d”,&no);
printf("\nEnter the strings:“);
for(i=0;i<no; i++)
    gets(string[i]);
```

```
for(i=0;i<no-1;i++)
  for(j=i+1;j<no;j++){
     if(strcmp(string[i],string[j])>0)
     {
     strcpy(temp,string[i]);
     strcpy(string[i],string[j]);
     strcpy(string[j],temp);
     }
    }

printf("\nThe sorted array is:“);
for(i=0;i<no;i++)
    puts(string[i]);
return 0;
}
```

Finding Substring in Main String

Main string: a cat is a cat that is a cat
Sub-String : cat

```
int main() {
 int i=0,j=0,k=0,count=0;
 int l=0,k1=0,cn[10],c=0;
 char a[80],b[80];

 printf("\nEnter main string:-\n“);
 gets(a);
 printf("\nEnter sub-string:-\n“);
 gets(b);
 l=strlen(b); //length of substring
```

```
Enter main string:- a cat is a cat that is a cat
Enter sub-string : cat
Substring is present 3 time(s) at position(s) 3   12   26
```

Finding Substring in Main String

```
Enter main string:- a cat is a cat that is a cat
Enter sub-string : cat
Substring is present 3 time(s) at position(s) 3   12   26
```

```
while (a[i]!='\0')  { //outer loop for MS
    if (a[i]==b[j]) {
            i++;
            j++;
            k1=1; //character match flag
            if (j==l) { //check for all chars match
               cn[c++]= i – l + 1; //pos array

                       //with occurrence count in 'c'
               j=0;
               k=1; //presence flag (SS)
              }
      }
```

```
else {
      if (k1==1){
             j=0;
             k1=0; //flag reset
            }
      else
            i++;
    }
 } //end of while
```

Finding Substring in Main String

```
Enter main string:- a cat is a cat that is a cat
Enter sub-string : cat
Substring is present 3 time(s) at position(s) 3   12   26
```

```
if (k==1) {

printf("\nSubstring is present  %d time(s) at  position(s)\t“, c);
  for(i=0;i<c;i++)
     printf(“%d\t”,cn[i]);
  }
 else   {
     if (k==0)
     printf("\nGiven sub-string is not present in the main string.”);
  }
return 0
} //end of program
```

Deleting repeated words in a sentence

```
Enter the string:- a cat is a cat that is a cat
String after deletion of repeated words: a cat is that
```

Tutorials on Strings

- Write a simple C program to retrieve first word from a sentence.
- Write a C program to remove blank space from the string
- Write a C program to count the number of vowels and consonants in a given string.
- Deleting repeated words in a given sentence.

Summary

- Strings definition, declaration, initialization
- Reading Strings
- String Handling Functions
- Programs using strings
- Array of Strings
- Operations on array of strings