COMPUTER PROGRAMMING AND UTILISATION
UNIT 4 – ARRAYS AND STRINGS

Introduction

- Array is a collection of variables of same type known by a single name.
- The array is an important concept and helps the programmer in managing many variables of the same type, because all the elements of an array share a single name.
- Arrays can be divided into two categories
    1. Single dimensional array
    2. Multi-dimensional array

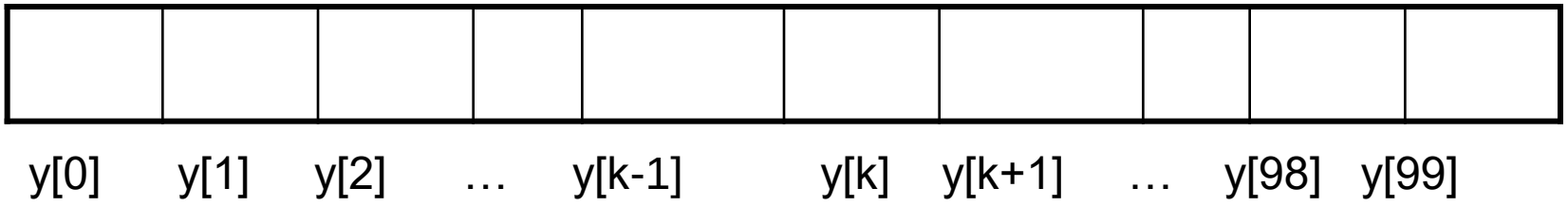
One-Dimensional Arrays

- Suppose, you need to store years of 100 cars. Will you define 100 variables?

```
int y1, y2,…, y100;
```

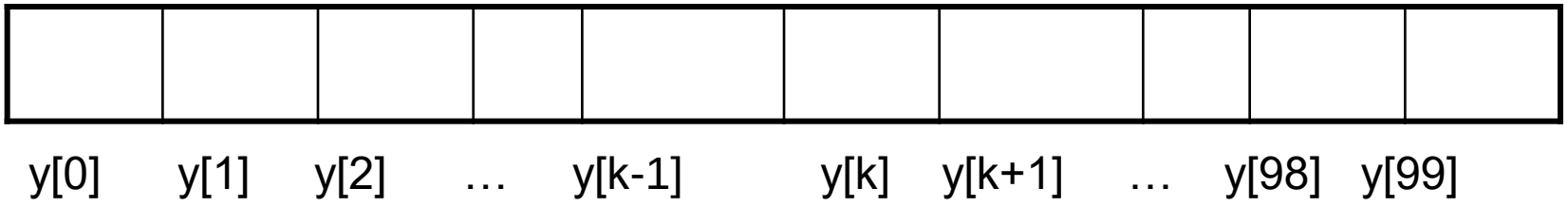
- An array is an indexed data structure to represent several variables having the same data type:

```
int y[100];
```



One-Dimensional Arrays (cont'd)

- An *element* of an array is accessed using the **array name** and an **index or subscript**, *for example:* `y[5]` which can be used like a variable
- In C, the subscripts always start with 0 and increment by 1, so `y[5]` is the sixth element
- The name of the array is the **address** of the first element and the subscript is the **offset**



Definition and Initialization

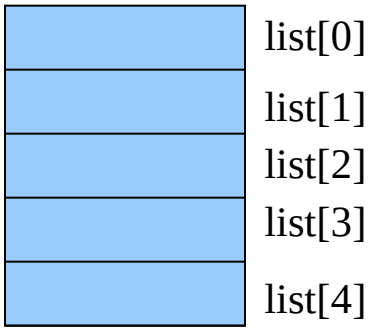
- An array is defined using a declaration statement.

```
data_type array_name[size];
```

  - allocates memory for `size` elements
  - subscript of first element is 0
  - subscript of last element is `size-1`
  - `size` **must be a constant**

Example


```
int list[5];
```



- allocates memory for 5 integer variables
- subscript of first element is 0
- subscript of last element is 4
- list[6] =5; /* may give segmentation fault or overwrite other memory locations*/

Initializing Arrays

- Arrays can be initialized at the time they are declared.
- Examples:

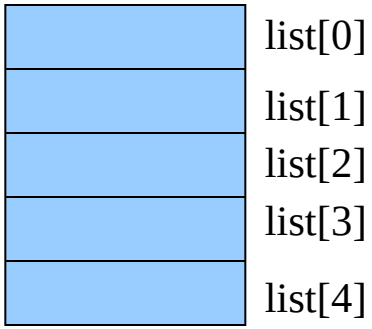
```
double taxrate[3] ={0.15, 0.25, 0.3};
char list[5] = {'h', 'e', 'l', 'l', 'o'};

double vector[100] = {0.0};
/* assigns zero to all 100 elements */

int s[] = {5,0,-5}; /*the size of s is 3*/
```

Assigning values to an array

For loops are often used to assign values to an array

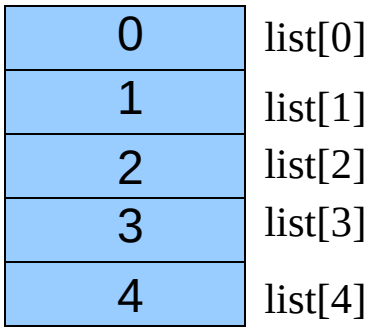
Example:

```
int list[5], i;
for(i=0; i<5; i++){
      list[i] = i;
}
```



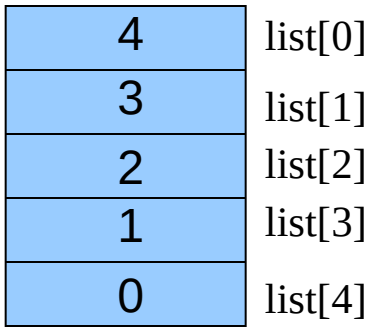
OR

```
for(i=0; i<=4; i++){
    list[i] = i;
}
```



Assigning values to an array

Give a for loop to assign the below values to list




```
int list[5], i;
for(i=0; i<5; i++){
      list[i] = 4-i;
}
```

Single Dimensional Array

- The syntax for declaring a single dimensional array is :
- **datatype arrayname[size];**
- Data type can be int, char, float , long int etc. The arrayname is the name of a variable which represents the array, while size which is represented in [ ] symbols represents the size of array.
- For example, **int a[10];**
- In 'C' language array index or subscript starts from 0. where a[0] is the first element of an array, while a[9] is the last element.

Program – sum and average of numbers using array

```
#include <stdio.h>
    main()
    {
            int a[10];    /* array of size 10 defined */
            int i,n;
            float avg, sum =0;   /* sum initialized to 0 */
            printf("Give value of n (not more than 10): ");
            scanf("%d",&n);  /* actual array size in n */
            for(i=0;i<n;i++)
            {
                    printf("Give number\n");
                    scanf("%d", &a[i]);      /* store array elements in array a */
                    sum = sum + a[i];      /* go on adding array element to sum */
            }
```

Program – sum and average of numbers using array

```
        avg = sum/n;   /* sum over. Calculate average */
        printf("Array elements are :\n");
        for (i=0; i<n;i++)     /* print array elements */
              printf("%d",a[i]);
        printf("\nSum = %f Average = %f\n", sum, avg); /* print answer */
}
```

Program – to find smallest and largest using array

```
#include <stdio.h>
#include <conio.h>
    main()
    {
            int a[10];    /* array of size 10 defined */
            int i,n;
            int max, min;
            clrscr();
            printf("Give value of n (not more than 10)\n");
            scanf("%d", &n);  /* actual array size in n */
            for(i=0;i<n;i++)
            {
                    printf("Give number\n");
                    scanf("%d", &a[i]);
            }
```

Program – to find smallest and largest using array (cont)

```
        max = a[0];      /* initialize min and max */
        min = a[0];
        for (i=1; i<n;i++)
        {
                if (max < a[i])
                        max = a[i];
                if (min > a[i])
                        min = a[i];
        }
         printf("Array elements are :\n");
         for (i=0; i<n;i++)
              printf("%d",a[i]);
        printf("\nLargest = %d Smallest = %d\n", max, min);
}
```

Program – count odd and even numbers

```
#include <stdio.h>
#include <conio.h>
    main()
    {
            int a[10];    /* array of size 10 defined */
            int i, n;
            int odd=0,even=0;  /* initialize counts */
            clrscr();
            printf("Give value of n (not more than 10)\n");
            scanf("%d", &n);  /* actual array size in n */
```

Program – count odd and even numbers (cont)

```
        for(i=0;i<n;i++)
        {
                printf("Give number\n");
                scanf("%d", &a[i]);
                if (a[i] %2 == 0)
                        even++;    /* increment even count */
                else
                        odd++;    /* increment odd count */
        }
        printf("Array elements are :\n");
        for (i=0; i<n;i++)
             printf("%d",a[i]);
        printf("\nNumber of ODDS = %d EVENS = %d\n", odd, even);
}
```

Program to sort numbers in ascending and descending order

```
#include <stdio.h>
#include <conio.h>
    main()
    {
            int a[10];    /* array of size 10 defined */
            int i, j, n, temp;
            clrscr();
            printf("Give value of n (not more than 10)\n");
            scanf("%d", &n);  /* actual array size in n */
            for(i=0;i<n;i++)   /* input data */
            {
                    printf("Give number\n");
                    scanf("%d", &a[i]);
            }
```

Program to sort numbers in ascending and descending order (cont)

```
        for(i=0;i<n-1;i++)
        {
                for(j=i+1;j<n;j++)
                {
                        if (a[i]> a[j]) /* exchange two numbers */
                        {
                                temp = a[i];  a[i] = a[j]; a[j] = temp;
                        }
                }
        }
        printf("Ascending order data is :\n");
        for (i=0; i<n;i++)  /* print from first to last */
                printf("%d",a[i]);
        printf("\nDescending order data is :\n");
        for (i=n-1; i>=0;i—) /* print from last to first */
                printf("%d",a[i]);
}
```

Two dimensional array

- Sometimes we need to store data where more than one dimensions are involved, like sales information of a company, or for mathematical calculations we need to use matrix.
- The syntax for two-dimensional array is :

  datatype variablename [rowsize] [colsize];

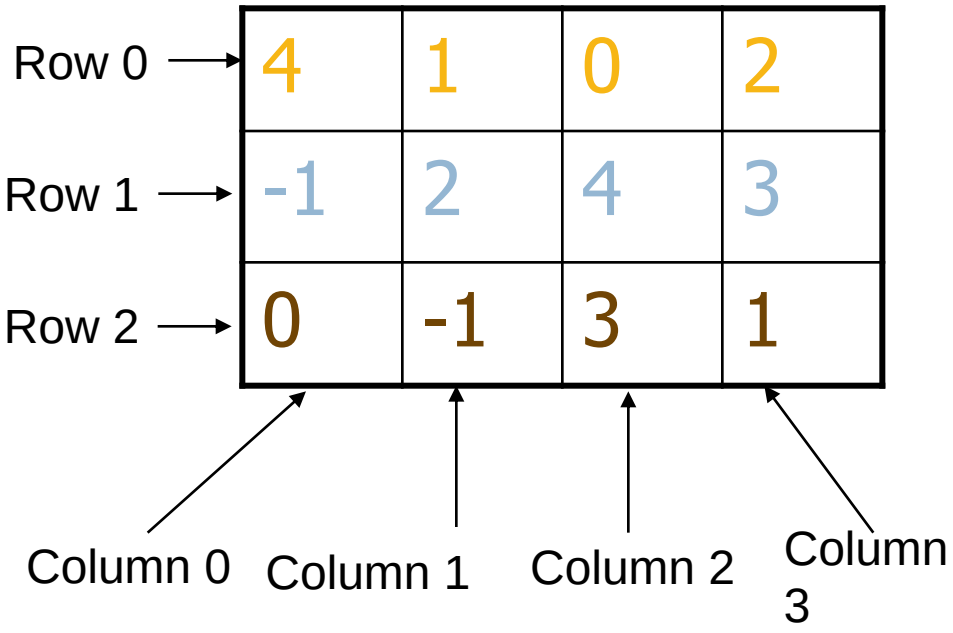
  where, **variablename** represents the name of an array, **rowsize** indicate the number of rows in table, **colsize** indicates number of columns in an array.

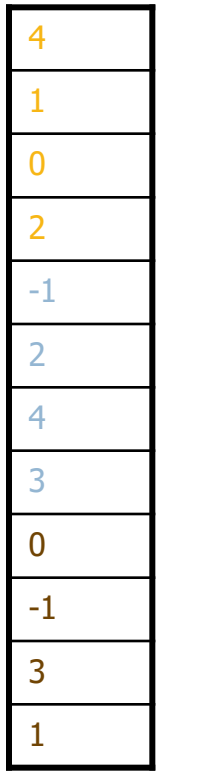
Matrices (2D-array)

	Column 0	Column 1	Column 2	Column 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

Matrices (2D-array)

- A matrix is a set of numbers arranged in a grid with rows and columns.
- A matrix is defined using a type declaration statement.
  - **`datatype array_name[row_size][column_size];`**
  - **`int matrix[3][4];`**





in memory

Accessing Array Elements

```
int matrix[3][4];
```

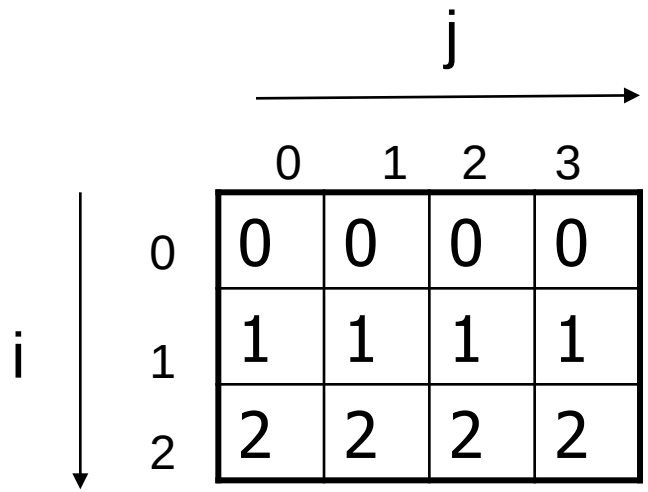
- `matrix` has 12 integer elements
- `matrix[0][0]` element in first row, first column
- `matrix[2][3]` element in last row, last column
- `matrix` is the address of the first element
- `matrix[1]` is the address of the Row 1
- `matrix[1]` is a one dimensional array (Row 1)

Initialization

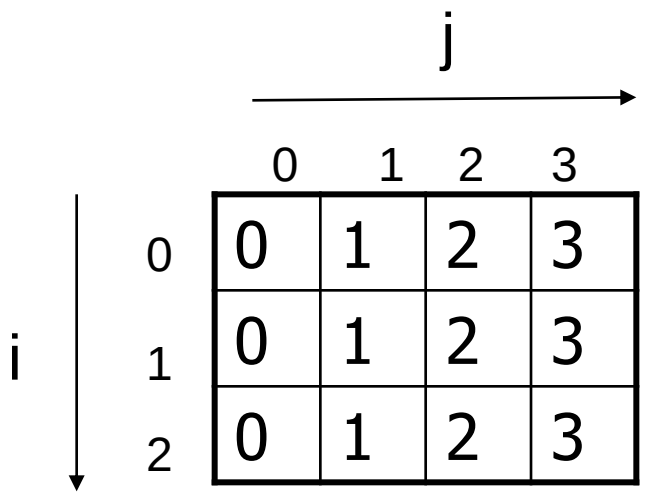
```
int x[4][4] = {   {2, 3, 7, 2},
                  {7, 4, 5, 9},
                  {5, 1, 6, -3},
                  {2, 5, -1, 3}};
int x[][4] = {    {2, 3, 7, 2},
                  {7, 4, 5, 9},
                  {5, 1, 6, -3},
                  {2, 5, -1, 3}};
```

Initialization

```
int i, j, matrix[3][4];
for (i=0; i<3; i++)
   for (j=0; j<4; j++)
     matrix[i][j] = i;
```




```
matrix[i][j] = j;
```



Program – to add two matrices

```
#include <stdio.h>
#include <conio.h>
    main()
    {
            int a[4][4], b[4][4],c[4][4];  /* matrix c stores result i.e c = a +b  */
            int m,n,p,q;                   /* m no.of rows in a, n no. of cols in a*/
            int i,j;  /* p no.of rows in b, q no. of cols in b*/
            clrscr();
            printf("Give number of rows in first matrix\n");
            scanf("%d", &m);
            printf("Give number of columns in first matrix\n");
            scanf("%d", &n);
            printf("Give number of rows in second matrix\n");
            scanf("%d", &p);
            printf("Give number of columns in second matrix\n");
            scanf("%d", &q);
```

Program – to add two matrices (cont)

```
if( m!= p || n!= q) /* check size match or not */
{
        printf("Size do not match. Addition not possible\n");
        return 0;
}
printf("Enter matrix A row-wise\n");
for(i=0;i<m;i++)   /* Get first matrix data */
{
        for(j=0;j<n;j++)
        {
                printf("a[%d][%d]=  ",i,j);
                scanf("%d",&a[i][j]);
        }
        printf("\n");
}
```

Program – to add two matrices (cont)

```
printf("Enter matrix B row-wise\n");
for(i=0;i<p;i++)   /* Get second matrix data */
{
        for(j=0;j<q;j++)
        {
                printf("b[%d][%d]=  ",i,j);
                scanf("%d", &b[i][j]);
        }
        printf("\n");
}
printf("Matrix A \n");
for(i=0;i<m;i++)          /* display first matrix row-wise */
{
        for(j=0;j<n;j++)
                printf("%d", a[i][j]);
        printf("\n");
}
```

Program – to add two matrices (cont)

```
        printf("Matrix B \n");
        for(i=0;i<p;i++)  /* display second matrix row-wise */
        {
                for(j=0;j<q;j++)
                        printf("%d", b[i][j]);
                printf("\n");
        }
        printf("\nSum Matrix C \n");
        for(i=0;i<m;i++)  /* calculate sum and display result matrix */
        {
                for(j=0;j<n;j++)
                {
                        c[i][j] = a[i][j] + b[i][j];
                        printf("%d", c[i][j]);
                }
                printf("\n");
        }
}
```

Program – Transpose matrix

```
#include <stdio.h>
#include <conio.h>
main()
{
        int a[4][4];
        int n;                          /* assuming rows and columns same i.e n *n matrix*/
        int i, j;
        clrscr();
        printf("Give value of n\n");
        scanf("%d", &n);
        printf("Enter matrix A row-wise\n");
        for(i=0; i<n; i++)   /* Get first matrix data */
        {
                for(j=0;j <n; j++)
                {
                        printf("a[%d][%d]=  ",i, j);
                        scanf("%d", &a[i][j]);
                }
                printf("\n");
        }
```

Program – Transpose matrix (cont)

```
        printf("Matrix A \n");
        for(i=0;i<n;i++)        /* display first matrix row-wise */
        {
                for(j=0;j<n;j++)
                        printf("%d", a[i][j]);
                printf("\n");
        }
        printf("Transpose of A \n");
        for(j=0;j<n;j++)        /* display second matrix row-wise */
        {
                for(i=0;i<n;i++)
                        printf("%d", a[i][j]);
                printf("\n");
        }
}
```

About Matrix Multiplication

- The number of columns of first matrix and number of rows of second matrix must be equal.
- When multiplying matrices, the elements of the rows in the first matrix are multiplied with corresponding columns in the second matrix.

$$\mathbf{A} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

$$\mathbf{AB} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} 1 \times a + 2 \times c & 1 \times b + 2 \times d \\ 3 \times a + 4 \times c & 3 \times b + 4 \times d \end{pmatrix} = \begin{pmatrix} a + 2c & b + 2d \\ 3a + 4c & 3b + 4d \end{pmatrix}$$

Matrix Multiplication

double a[3][2], b[2][4], c[3][4];

- Find c = a * b;

3	4
5	2
1	6

x

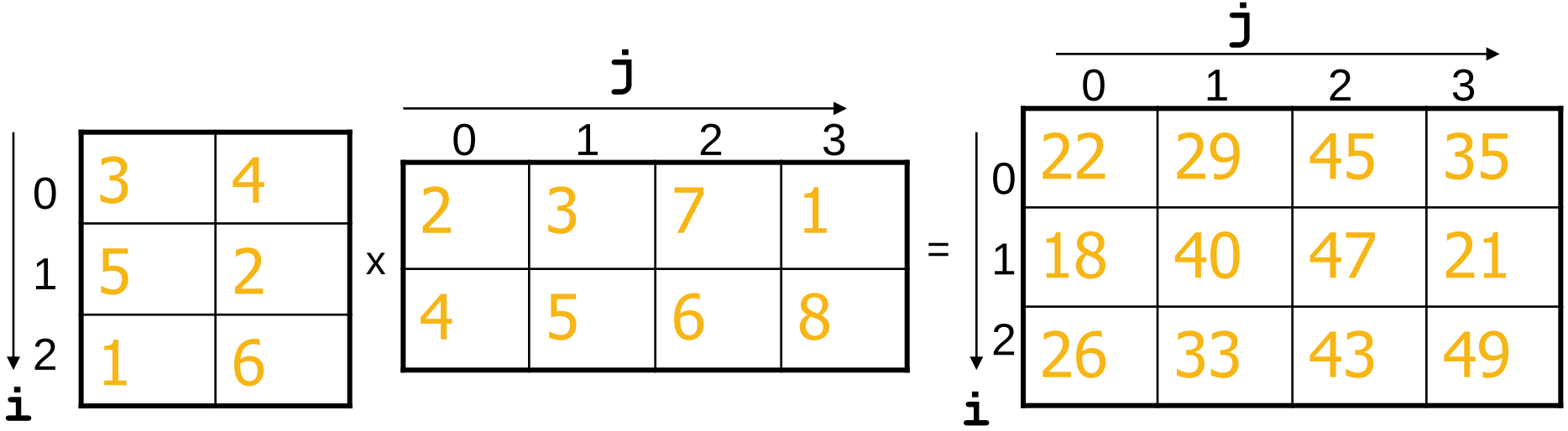
2	3	7	1
4	5	6	8

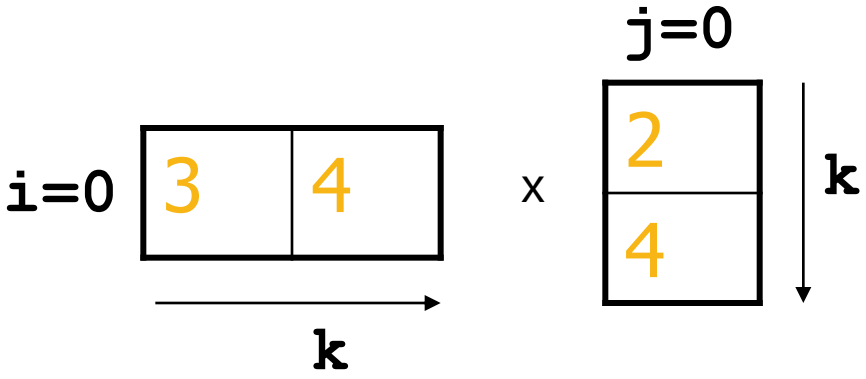
=

22	29	45	35
18	40	47	21
26	33	43	49

3*2 + 4*4=22	3*3 + 4*5=29	3*7 + 4*6=45	3*1 + 4*8=35
5*2 + 2*4=18	5*3 + 2*5=40	5*7 + 2*6=47	5*1 + 2*8=21
1*2 + 6*4=26	1*3 + 6*5=33	1*7 + 6*6=43	1*1 + 6*8=49

Matrix Multiplication cont'd





=

```
c[i][j] =
  a[i][k=0]*b[k=0][j] +
  a[i][k=1]*b[k=1][j]
```

Matrix Multiplication cont'd

2	3	1
4	0	2
3	5	1

1	2	0
4	1	3
4	2	1

18	9	10
12	12	2
27	17	8

Matrix Multiplication cont'd

```
c[0][0] = a[0][0]*b[0][0] + a[0][1]*b[1][0] + a[0][2]*b[2][0]
c[0][1] = a[0][0]*b[0][1] + a[0][1]*b[1][1] + a[0][2]*b[2][1]
c[0][2] = a[0][0]*b[0][2] + a[0][1]*b[1][2] + a[0][2]*b[2][2]

c[1][0] = a[1][0]*b[0][0] + a[1][1]*b[1][0] + a[1][2]*b[2][0]
c[1][1] = a[1][0]*b[0][1] + a[1][1]*b[1][1] + a[1][2]*b[2][1]
c[1][2] = a[1][0]*b[0][2] + a[1][1]*b[1][2] + a[1][2]*b[2][2]

c[2][0] = a[2][0]*b[0][0] + a[2][1]*b[1][0] + a[2][2]*b[2][0]
c[2][1] = a[2][0]*b[0][1] + a[2][1]*b[1][1] + a[2][2]*b[2][1]
c[2][2] = a[2][0]*b[0][2] + a[2][1]*b[1][2] + a[2][2]*b[2][2]
```

Crux Portion…

```
for(i = 0 ; i < row1 ; i++)
{
    for(j = 0 ; j < col2 ; j++)
    {
        c[i][j] = 0 ;
        for(k = 0 ; k < col1 ; k++)
            c[i][j] = c[i][j] + a[i][k] * b[k][j] ;
    }
}
```

Matrix Multiplication cont'd

```
#define N  3
#define M  2
#define L  4
void matrix_mul(a[N][M], int b[M][L], int c[N][L])
{
  int i, j, k;
  for(i=0; i < N; i++) {
     for(j=0; j < L; j++) {
         c[i][j] = 0;
         for(k=0; k < M; k++) {
            c[i][j] = c[i][j] + a[i][k] * b[k][j];
         }
     }
  }
  return;
}
```

Exercise

- Write the nested loop to initialize a 2D array as follow

0	1	2
1	2	3
2	3	4
3	4	5

```
int i, j, x[4][3];
for(i=0; i<4; i++)
   for(j=0; j<3; j++)
      x[i][j] = i+j;
```

String

- String is a sequence of characters enclosed in double quotes. ASCII code is internally used to represent string in memory.
- In 'C' each string is terminated by a special character called null character is represented as '\0' or NULL.
- Because of this reason, the character array must be declared one size longer than the string required to be stored.
- String is basically an array of characters, so we can initialize the string by using the method of initializing the single dimensional array as shown below.

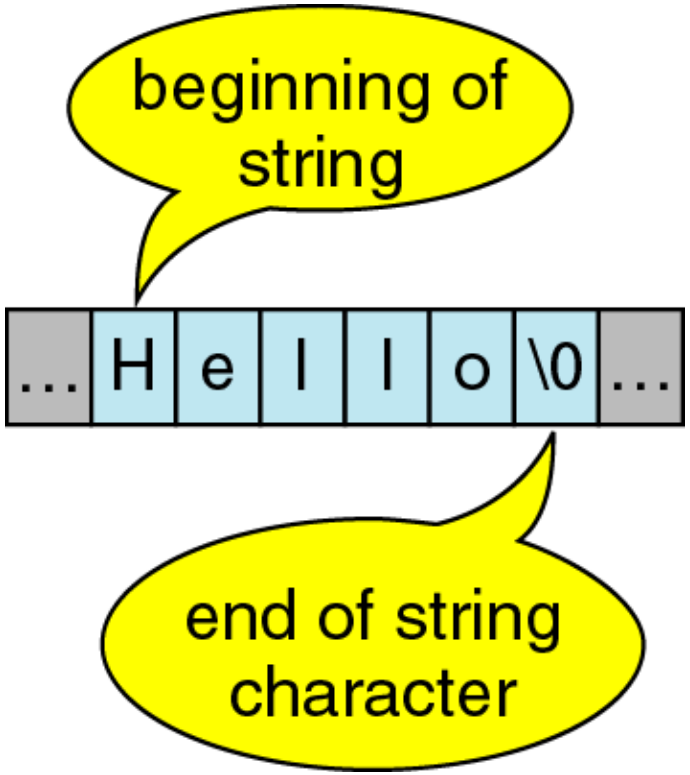
char name[] = { 'C', 'O' , 'M', 'P', 'U', 'T','E','R','\0'};

or

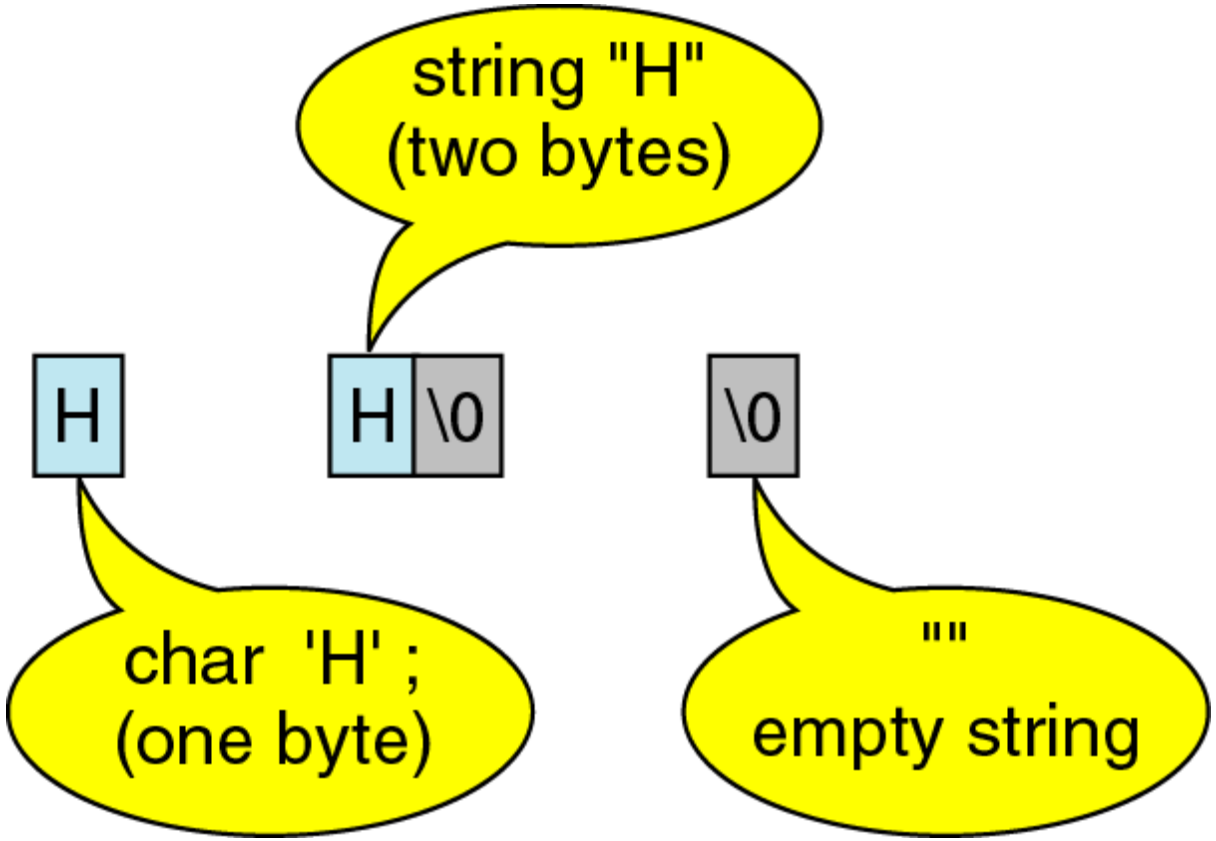
char name[] = "COMPUTER";

- When the string is written in double quotes, the NULL character is not required, it is automatically taken.

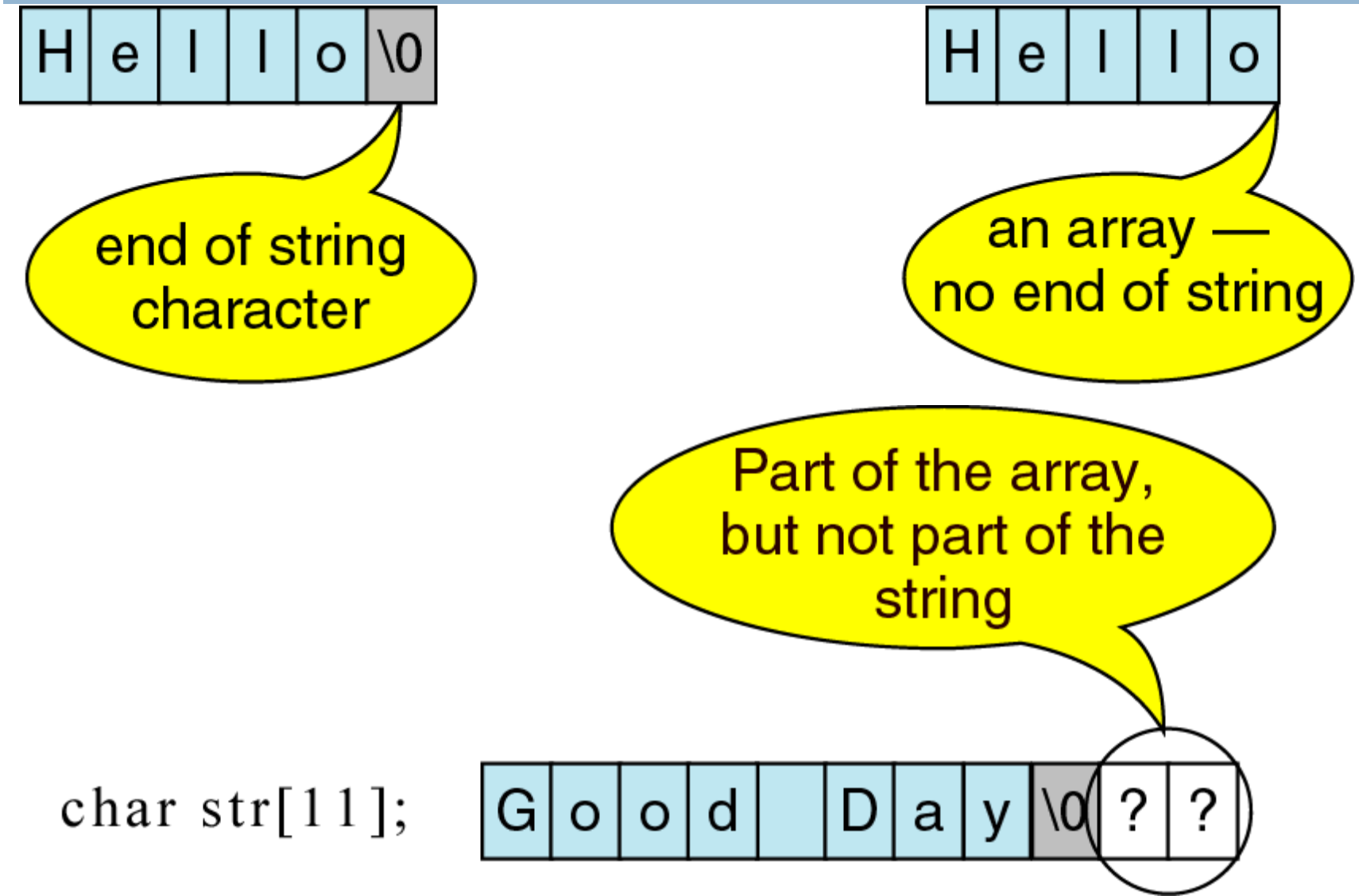
String



String



String

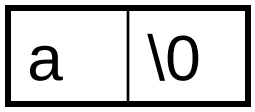

H e l l o \0
end of string character
H e l l o
an array — no end of string
Part of the array, but not part of the string
char str[11];
G o o d D a y \0 ? ?

Character vs. String

- A string constant is a sequence of characters enclosed in double quotes.
  - For example, the character string:

```
char s1[2]="a";  //Takes two bytes of storage.
```

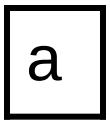
s1:



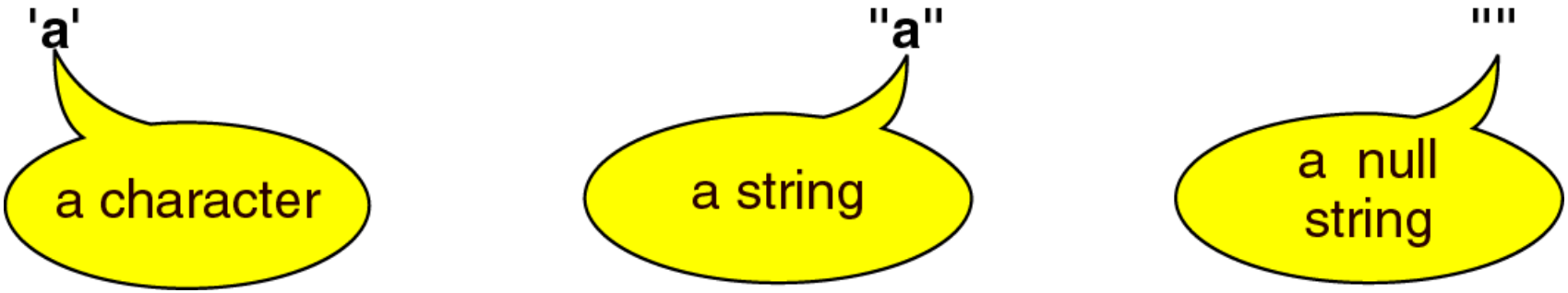
  - On the other hand, the character, in single quotes:

```
char s2= `a`; //Takes only one byte of storage.
```

s2:



Character vs. String



Reading strings

- The format specifier %s is used to read a string in the character array with scanf() statement.
- We can also use the function **gets()** for reading a string.
- The array name itself works as a pointer to the first character. So & is not required to be used before array name.
- The advantage of gets() is that we can read strings involving blanks, tabs.
- While the scanf() reads only upto blank or tab character. So, scanf() is used to read word while gets() can be used read sentence involving many words.

Printing strings

- The string can be printed using printf() function with %s format specifier, or by using puts() function

Read string from terminal.

```
#include <stdio.h>
int main()
   {
    char name[20];
    printf("Enter name: ");
    scanf("%s", name);
    printf("Your name is %s.", name);
    return 0;
    }
```

Read line of text character by character

```
#include <stdio.h>
int main()
{
    char name[30], ch;
    int i = 0;
    printf("Enter name: ");
    while(ch != '\n')    // terminates if user hit enter
    {
        ch = getchar();
        name[i] = ch;
        i++;
    }
    name[i] = '\0';       // inserting null character at end
    printf("Name: %s", name);
    return 0;
}
```

Read line of text using gets() and puts()

```
#include <stdio.h>
int main()
{
    char name[30];
    printf("Enter name: ");
    gets(name);      //Function to read string from user.
    printf("Name: ");
    puts(name);     //Function to display string.
    return 0;
}
```

String

Program

```
main()
{
    char country[15] = "United Kingdom";
    printf("\n\n");
    printf("*123456789012345*\n");
    printf(" ------ \n");
    printf("%15s\n", country);
    printf("%5s\n", country);
    printf("%15.6s\n", country);
    printf("%-15.6s\n", country);
    printf("%15.0s\n", country);
    printf("%.3s\n", country);
    printf("%s\n", country);
    printf("------ \n");
}
```

Output

```
*123456789012345*
------
United Kingdom
United Kingdom
      United
United
Uni
United Kingdom
------
```

Program – to find length of a given string

```
#include <stdio.h>
#include <conio.h>
main()
{
        char str[30];    /* array of size 30 defined */
        int i, count =0;  /* initialize count to zero */
        clrscr();
        printf("Give string \n");
        gets(str);
        for(i=0; str[i] != NULL; i++)
                count++;              /* Increment count */
        printf("Length of string %s = %d\n", str, count);
}
```

Program – read string and print ASCII code of each character

```
#include <stdio.h>
#include <conio.h>
main()
{
        char str[30];    /* array of size 30 defined */
        int i;
        clrscr();
        printf("Give string \n");
        gets(str);
        for(i=0;i< str[i] != NULL;i++)
                printf("%c = %d\n", str[i], str[i]);
}
```

Program – to reverse a string

```
#include <stdio.h>
#include <conio.h>
main()
{
        char str[30], rev[30];/* array of size 30 defined */
        int i, j, count=0;   /* initialize count to zero */
        clrscr();
        printf("Give string \n");
        gets(str);
        for(i=0;i< str[i] != NULL;i++)          /* Get length of string in count */
                count++;
        for(i=count-1,j=0;i>=0;i—,j++)
        /* Last to first character stored in array rev */
                rev[j] = str[i];
        rev[j] = NULL;   /* Terminate the reverse string with NULL */
        printf("Original string %s\t Reverse string %s\n", str, rev);
}
```

Example - 1

```
void main()
{
char text[15]="MOUNTAIN";
int i;
clrscr();
for(i=0;i<=7;i++)
printf("%c ",text[i]+2);
getche();
}
```

Example - 2

```
void main()
{
char text[15]="ZYX";
int i;
clrscr();
for(i=0;i<=2;i++)
printf("%d ",text[i]-2);
getche();
}
```

Example - 3

```
#include <string.h>
void main()
{
char alpha='z';
int number=50;
char text1[20]="Pleasent";
clrscr();
alpha='z'-25;
printf("\n%c %d", alpha,'z'+1);
printf("\n%d %c", number,
    number*2);
printf("\n%s ", text1+4);
getche();
}
```

Example - 4

```
#include <string.h>
void main()
{
char alpha='z';
int number=67;
char text1[20]="Discovery";
clrscr();
alpha='z'-1;
printf("\n%c %d", alpha,'z'+1);
printf("\n%d %c",
    number+2,number-2);
printf("\n%s ", text1+5);
getche();
}
```

Example - 5

```
void main()
{
char s;
clrscr();
for(s=97;s<=122;s++)
{
if(s>100 && s<123)
continue;
printf("\n%c %d",s,s);
}
getch();
}
```

String handling built-in functions

- We need to include string.h header file for using built-in string functions.

Name	Syntax	Meaning of function
strlen	strlen(s)	Finds the length of string s.
strcpy	strcpy(dest,src)	Copies the string src to dest.
strcat	strcat(s1,s2)	Concate string s2 at the end of string s1.
strcmp	strcmp(s1,s2)	Compares string s1 with s2. If both are equal, it returns 0. If s1 alphabetically > s2, it returns positive number, otherwise returns negative number.
strrev	strrev(s)	Reverses the string s, the original string is overwritten.
strcmpi	strcmpi(s1,s2)	Compares string s1 with s2 **ignoring the case.** If both are equal, it returns 0. If s1 alphabetically > s2, it returns positive number, otherwise returns negative number.

String handling built-in functions (cont)

Name	Syntax	Meaning of function
strncmp	strncmp(s1,s2,n)	Compare first n characters of s1 and s2 and return result similar to strcmp
strupr	strupr(s)	Convert string s to uppercase
strlwr	strlwr(s)	Convert string s to lowercase
strstr	strstr(s1,s2)	Returns a pointer to the first occurrence of string s2 in s1

Program – print every third character if lowercase

```
#include<stdio.h>
#include<string.h>
main()
{
        char s[20];
        int i,len;
        printf("Give one string\n");
        gets(s);
        len = strlen(s);
        for(i=2;i<len; i=i+3)
        {
                if (islower(s[i])) /* is the character lowercase ? */
                        printf("%c", s[i]);
        }
}
```

Program – convert string to upper and lower case

```
/*    ASCII value of A = 65, B=66,…Z=90 ,     ASCII value of a = 97, b=98,…z=122
    So, difference between ASCII value of second  alphabet and first alphabet is 97-65 =32 */
    #include <stdio.h>
    #include <conio.h>
    main()
    {
            char str[20],ustr[20],lstr[20];          /* array of size 20 defined */
            int i,count=0;    /* count initialized to 0 */
            clrscr();
            printf("Give string \n");
            gets(str);        /* string in str array */
            for(i=0;i< str[i] != NULL;i++)
                        {
                        if ( str[i] >= 'a' && str[i] <='z') /* check for lower case */
                                    ustr[i] = str[i] - 32;       /* convert to upper */
                        else
                                    ustr[i] = str[i];
```

Program – convert string to upper and lower case (cont)

```
                    if ( str[i] >= 'A' && str[i] <='Z')  /* check for upper case */
                        lstr[i] = str[i] + 32;  /* convert to lower */
                else
                        lstr[i] = str[i];
                }
        lstr[i] = NULL; /* put NULL character at end in both array*/
        ustr[i] = NULL;
        printf("\nOriginal String is %s\n",str);
        printf("\Upper Case String is %s\n",ustr);
        printf("\Lower Case String is %s\n",lstr);
}
```

Program –Search a string

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
    main()
    {
            char names[10][20];
            char str[20];
            int i,n;
            int flag =0;
            clrscr();
            printf(“How many strings\n”);
            scanf(“%d”,&n);
            printf("Enter strings\n“);
            for(i=0;i<n;i++)
                        scanf("%s“,names[i]);
            printf(“Given strings are:\n”);
```

Program –Search a string (cont)

```
        for(i=0;i<n;i++)
                printf("%s\t",names[i]);
        printf("\nEnter the string you want to search\n");
        scanf("%s",str);
        for(i=0;i<n;i++)
        {
                if (strcmp(names[i],str) == 0)
                {
                        flag =1;
                        break;
                }
        }
        if (flag ==1)
                printf("Found !! Given string %s found in list\n",str);
        else
                printf("Sorry !! Given string %s not found in list\n",str);
}
```