This lecture covers

- **Arrays and Matrices:**
  - One-Dimensional Array
  - Sorting Algorithms
  - Search Algorithms
  - Two-Dimensional Arrays

To Understand Basic Concepts of Arrays

- Consider a scenario, which requires to read, process and print scores of 10 candidates.
- In order to store the 10 scores , 10 integer variables need to be declared :
  int score1, score2, score3, score4, score5, score6, score7, score8, socre9, score10;
- The above declaration arises lots of ambiguity in declaring the variable names which may be acceptable for 10 scores but definitely not acceptable for 100,1000,10000 scores

Why we need an Array

- To process large amount of data we need a powerful data structure
- A data structure specifies how elements are arranged and organized in the memory for easy access, for operations like insertion, deletion , updating
- Variables of simple data types, such as int, float, and char, can hold only one value at any time during program execution, although that value may change
- To process collection of elements we need derived data type which can hold multiple values at the same time

Definition of Array

Array is a data structure which collects related data items that share a common array name and data type is called an array.

Properties of an Array

- An array's data items are stored contiguously in memory.
- An array is a group of homogeneous data items that all have the same array name and same data type.
- The data type of array is chosen as per the requirement
- Arrays are static in that they remain the same size throughout program execution
- Each of the data items is known as an element of the array
- Each element can be accessed individually using index or subscript

For example we can define an array name marks to represent a set of marks obtained by a group of students. A particular value is indicated by writing a number called index number or subscript in brackets after the array name.

Example: Marks[7]
Represents the marks of the 7 students. The complete set of values is referred to as an array; the individual values are called elements. The arrays can be of any variable type.

One-dimensional array:

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

int y1, y2,…, y100; [**Answer : No]**

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

When a list of items can be given one variable name using only one subscript and such a variable is called a single-subscripted variable or one dimensional array. In C language, single-subscripted variable x_i can be represented as

x[1],x[2],x[3]……x[n]

The subscripted variable x_i refers to the i^{th} element of x. The subscript can begin with number 0. For example, if we want to represent a set of five numbers, say (57, 20, 56, 17, 23), by an array variable num, then we may declare num as follows

int num[5];

And the computer reserves five storage locations as shown below:

Num[0]
Num[1]
Num[2]
Num[3]
Num[4]

The values can be assigned as follows:

Num[0]=57;
Num[1]=20;
Num[2]=56;
Num[3]=17;
Num[4]=23;

The table below shows the values that are stored in the particular numbers.

Num[0]
Num[1]

Num[2]
Num[3]
Num[4]
57
20
56
17
23

What is a multidimensional array?

programming language allows programmer to create arrays of arrays known as multidimensional arrays.

For example: float a[2][6];

Here, *a* is an array of two dimension, which is an example of multidimensional array. This array has 2 rows and 6 columns.

Now let us discuss the "advantages and disadvantages of arrays?"

Advantages:

- It is used to represent multiple data items of same type by using only single name.
- It can be used to implement other data structures like linked lists, stacks, queues, trees, graphs etc.
- 2D arrays are used to represent matrices.

Disadvantages:

- We must know in advance that how many elements are to be stored in array.
- Array is static structure. It means that array is of fixed size. The memory which is allocated to array cannot be increased or reduced.
- Since array is of fixed size, if we allocate more memory than requirement then the memory space will be wasted. And if we allocate less memory than requirement, then it will create problem.
- The elements of array are stored in consecutive memory locations. So insertions and deletions are very difficult and time consuming.

Question: Differentiate between i++ and i--?

- Pre-increment ++I increments the value of I and evaluates to the new incremented value

int i = 3;

```
int preIncrementResult = ++i;
printf("The value of I is %d", preIncrementResult);
```

- Post-increment I++ increments the value of I and evaluates to the original non incremented value

```
int i = 3;
int postIncrementResult =i++;
printf("The value of I is %d", postIncrementResult);
```

Sorting and Searching Techniques

Bubble Sort:

What is sorting?

The process of arranging the given elements so that they are in ascending order or descending order is called sorting.

For example: Consider the unsorted elements
10, 50, 25, 15
After sorting in ascending order, we get the following
10,15, 25, 50

1. Write a program to sort the arrays in ascending order using bubble sort:

```
#include<stdio.h>
#include<conio.h>
void bubble_sort(int a[], int n)
{
     int i, j, temp;
    for(j=1; j<n; j++)
   {
      for(i=0; i<n-j; i++)
        {
         if(a[i]>=a[i+1])
          {
              temp=a[i];
              a[i]=a[i+1];
              a[i+1]=temp;
          }
        }
    }
}
void main()
{
    int a[20],b[20],n, i;
```

```
    printf("Enter the number of elements");
    scanf("%d", &n);
    printf("Enter the unsorted elements");
      for(i=0; i<n; i++)
       {
         scanf("%d", a[i]);
          b[i]=a[i];
       }
     bubble_sort(a, n);
     printf("Given Array is:");
   for(i=0;i<n;i++)
   {
    printf("%d, %d",b[i], a[i]);
    }
}
```

Advantages of Bubble sort:

- Very simple and easy program.
- Straight forward approach.

Disadvantages of bubble sort:

- It runs slowly and hence it is not efficient. More efficient sorting techniques are present.
- Even if the elements are sorted, n-1 passes are required to sort.

Searching:

What is searching?

The process of finding a particular element/item in the large amount of data is called searching.

Two important searching techniques are:

- Linear search
- Binary search

❖ **Linear search (sequential search):**

In this technique we search for a given *key* item in the list in linear order i.e one after the other from first element to last elements or vice versa.

The search may be successful or unsuccessful. If key item is present, we say search is successful, otherwise search is unsuccessful.

2. Write a program to implement linear search.

```
#include<stdio.h>
#include<conio.h>

int linear (int key, int a[], int n)
{
   int i;
  for(i=0;i<n;i++)
  {
     if(key==a[i])
      return i;
    }
   return -1;
}
void main()
{
   int n, key, a[20], i, pos;
   printf("enter the value of n");
   scanf("%d", &n);
   prinf("enter the elements of array");
   for(i=0; i<n;i++)
   scanf("%d", &a[i]);
   printf("enter the item to be searched");
   scanf("%d", &key);
   pos=linear(key, a, n);

    if(pos==-1)
   printf("Item not found");
   else
   printf("Item found\n");
}
```

Advantages of Linear search

- Very simple approach
- Works well for small array.
- Used to search when the elements are not sorted.

Disadvantages of Linear search

- Less efficient if the array size is large.
- If the elements are already sorted, linear search is not efficient.

❖ **Binary search:**

A binary search is a technique which can be applied if the items to be compared are either in ascending / descending order.

3. Write a program to implement binary search using function.

```
#include<stdio.h>
#include<conio.h>
int binary_search(int key, int a[], int n)
{
    int low, high, mid;
      low=0;
     high=n-1;
   while(low<=high)
  {
   mid=(low+high)/2;

   if(key==a[mid])
      return mid;
  if(key<a[mid])
      high=mid-1;
  else
     low=mid+1;
  }
   return -1;
}

void main()
{
     int n, a[10], key, pos;
    printf("enter the no. of elements");
    scanf("%d", &n);
    printf("enter the elements");
  for(i=0;i<n;i++)
   {
     Scanf("%d",&a[i]);
   }
 prinf("enter the elements to be searched");
 scanf("%d", &key);
 pos=binary_search(key, a, n);
  if(pos==-1)
            printf("Item not found");
    else
            printf("Item found\n");
}
```

Advantages of binary search:

- Simple Technique
- Very efficient searching technique

Disadvantages of binary search:

- The list of elements to be searched should be sorted.
- It is necessary to find the middle elements to search any key in the given list

Two -Dimensional Arrays:

There are certain situations where a table of values will have to be stored. C allows us to define such table using two dimensional arrays.

Two dimensional arrays are declared as follows:

Syntax: Type array_name [row_size][column_size];

In c language the array sizes are separated by its own set of brackets.

Two dimensional arrays are stored in memory as shown in the table below. Each dimension of the array is indexed from zero to its maximum size minus one; the first index selects the row and the second index selects the column within that row.

	Column0 [0][0]	Column1 [0][1]	Column2 [0][2]
Row 0	210	340	560
	[1][0]	[1][1]	[1][2]
Row 1	380	290	321
	[2][0]	[2][1]	[2][2]
Row2	490	235	240
	[3][0]	[3][1]	[3][2]
Row3	240	350	480

Declaration and Initialization of Arrays

- **Declaration:**

  The arrays are declared before they are used in the program. The general form of array declaration is

type variable_name[size];

The type specifies the type of element that will be contained in the array, such as int, float,or char and the size indicates the maximum number of elements that can be stored inside the array.

Example: **1.** float weight[40];
Declares the weight to be an array containing 40 real elements. Any subscripts 0 to 39 are valid.

2. int group1[11];
Declares the group1 as an array to contain a maximum of 10 integer constants.

The C language treats character strings simply as arrays of characters. The size in a character string represents the maximum number of characters that the string can hold.

For example:

char text[10];

Suppose we read the following string constant into the string variable text.
"HOW ARE YOU"

Each character of the string is treated as an element of the array text and is stored in the memory as follows.

'H'
'O'
'W'
'A'
'R'
'E'
'Y'
'O'
'U'
'\o'

When the compiler sees a character string, it terminates it with an additional null character. Thus, the element text[11] holds the null character '\o' at the end. When declaring character arrays, we must always allow one extra element space for the null terminator.

Initialization of arrays
The general form of initialization of arrays is:

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

The values in the list are separated by commas.
For example, the statement below shows

static int num[3]={2,2,2};

Will declare the variable num as an array of size 3 and will assign two to each element. If the number of values is less than the number of elements, then only that many elements will be initialized. The remaining elements will be set to zero automatically.

For example:

static float num1[5]={0.1,2.3,4.5};

Will initialize the first three elements to 0.1,2.3 and 4.5 and the remaining two elements to zero. The word static used before type declaration declares the variable as a static variable. In some cases the size may be omitted. In such cases, the compiler allocates enough space for all initialized elements.

For example, the statement

static int count[ ]= {2,2,2,2};

Will declare the counter array to contain four elements with initial values 2.
Character arrays may be initialized in a similar manner. Thus, the statement

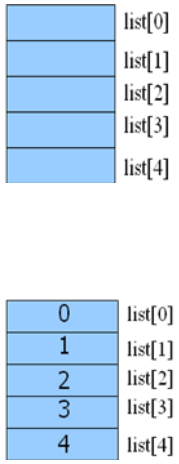
static char name[ ]={ 'S 'W,'A,'N}

Declares the name to be an array of four characters, initialized with the string "SWAN"
There certain draw backs in initialization of arrays.
There is no convenient way to initialize only selected elements.
There is no shortcut method for initializing a large number of array elements.

Assigning values to an array

for loops are often used to assign values to an array:

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



Using #define for Array Sizes

```
#define SIZE 39
#define GRADES 5
int main ( void )
{
   int score [ SIZE ] ;
   int gradeCount [ GRADES ] ;
                    –
                    –
                    –
}
```

Programming Examples

1. Program to read and write two dimensional array

```
#include<stdio.h>
main()
{
int a[10][10];
int i,j,row,col;
printf("\n Input row and column of a matrix:");
scanf("%d %d", &row,&col);
for(i=0; i<row;i++)
for(j=0;j<col;j++)
scanf("%d", &a[i][j]);
for(i=0;i<row;i++) {
for(j=0;j<col;j++)
printf("%5d", a[i][j]);
printf("\n");
}
```

2. Program showing one-dimensional array

```
main()
{
int i;
float a[10],value1,total;
printf("Enter 10 Real numbers\n");
for(i=0;i<10;i++)
{
scanf("%f", &value);
x[i]=value1;
}
total=0.0;
for(i=0;i<10;i++)
total=total+a[i]*a[i];
printf("\n");
for(i=0;i<10;i++)
printf("x[%2d]= %5.2f\n", i+1, x[i]);
printf("\ntotal=%.2f\n", total);
}
```

3. Program to print multiplication tables

```
#define R1 4
#define C1 4
main()
{
int row,col,prod[R1][C1];
int i,j;
printf(" MULTIPLICATION TABLE \n\n");
printf(" ");
for(j=1;j<=C1;j++)
printf("%4d",j);
printf("\n");
printf("----------------------------------------\n");
for(i=0;i<R1;i++)
{
row=i+1;
printf("%2d|", R1);
for(j=1;j<=C1;j++)
{
col=j;
prod[i][j]=row*col;
printf("%4d", prod[i][j]);
}
printf("\n"); } }
```

Output

MULTIPLICATION TABLE
1 2 3 4
---
1 | 1 2 3 4
2 | 2 4 6 8
3 | 3 6 9 12
4 | 4 8 12 16

4. Program to show swapping of numbers.

```
#include<stdio.h>
main()
{
int x[10], i,j,temp;
for(i=0;i<=9;++i)
scanf("%d", &x[i]);
for(j=0;j<=8;j+=2)
{
temp=x[j];
x[j]=x[j+1];
x[j+1]=temp;
}
for(i=0;i<=9;++i)
printf("%d", x[i]);
printf("\n");
}
```

5. Program to sort a list of numbers using bubble sort:

```
#define N 10
main()
{
int i,j,k;
float a[N],t;
printf("Enter the number of items\n");
scanf("%d", &n);
printf("Input %d values \n", n);
for(i=1; i<=n ;;i++)
scanf("%f", &a[i]);
for(i=1;i<=n-1;i++)
{
for(j=1;j<=n-i; j++)
{
if)a[j]<=a[j+1])
{
t=a[j];
a[j]=a[j+1];
a[j+1]=t;
}
else
continue;
}
```

```
}
for(i=1;i<=n;i++)
printf("%f", a[i]);
}
```

6. Program to calculate standard deviation

```
#include<math.h>
#define MAX 100
main()
{
int i,n;
float val[MAX],deviation,sum,ssqr,mean,var,stddev;
sum=ssqr=n=0;
printf("Input values: input-1 to end\n");
for(i=1;i<MAX;i++)
{
scanf("%f", &val[i]);
if(val[i]==-1)
break;
sum+=val[i];
n+=1;
}
mean=sum/(float)n;
for(i=1;i<=n;i++)
{
deviation=val[i]-mean;
ssqr+=deviation*deviation;
}
var=ssqr/(float)n;
stddev=sqrt(var);
printf("\n Number of items:%d\n",n);
printf("Mean: %f \n", mean);
printf("Standard deviation: %f\n", stddev);
}
```

7. Program to find the largest and smallest of numbers

```
#include<stdio.h>
main()
{
int i,s, largcount,smcount;
float num[30],lar,small;
printf("\n size of array (MAX 30): \t");
scanf("%d", &size);
printf("\n Array elements:\t");
for(i=0;i<size;i++)
scanf("%f", &num[i]);
for(i=0;i<size;i++)
printf("%f", &num[i]);
lar=small=num[0];
larcount=smcount=1;
for(i=1;i<size;i++)
{
if(num[i]>lar)
{
lar=num[i];
larcount=i+1;
}
elseif(num[i]<small)
{
small=num[i];
smcount=i+1;
}
}
printf("\n Largest value is % f found at %d", lar,larcount);
printf("\n Smallest value is %f found at %d ", small, smcount);
}
```

8. Design a program to find maximum value in data array where the elements in the array are random number between 0 - 99

```
#include <stdio.h>
{
int data[10], max, i;
    for (i=0; i<5; i++)
      data[i] = rand()%100;
    max = data[0];
  for (i=1; i<5; i++)
{
     if (data[i] > max)
        max = data[i];
  }
  printf("Max = %d\n",max);
return 0 ;
}
```

Output:

Consider 5 random numbers: 35, 4,78,12,45

Max = 78

9. Write a program to generate Fibonacci numbers using Arrays

```
void Fibonacci (int a[], int n)
{
int i;
a[0]=0;
a[1]=1;
/* generate other Fibonacci numbers*/
for(i=2;i<n;i++)
   {
    a[i]=a[i+1] +a[i+2];
    }
}
void main()
{
int n, i, a[100];
printf("Enter the value for n \n ");
scanf("%d", &n);
Fibonacci(a, n);
Printf(" The Fibonacci numbers are:");
for(i=0;i<n;i++)
  {
   printf("%d\n,a[i]"); } }
```

10. Write a program to add all the element of single dimensional array.

```
#include<iostream.h>
#include<conio.h>
int main()
{
        int Arr[100],n,sum=0;
        cout<<"Enter number of elements you want to insert ";
        cin>>n;

        for(int i=0;i<n;i++)
        {
                cout<<"Enter element "<<i+1<<":";
                cin>>Arr[i];
        }
    for(i=0;i<n;i++)
    sum+=Arr[i];
    cout<<"\nThe sum of Array is :"<<sum;
  cout<<"\nThe average of Array is :"<<sum/i;
  return 0;
}
```

11. Write a program to find largest and smallest array in an array.

```
#include <stdio.h> //include header file

int main () //start of main fcn
{

  int values[ 20 ];     //delcares array and how many elements
  int small,big;        //declares integer
  big=small=values[0];  //assigns element to be highest or lowestvalue


   for ( int i = 0; i < 20; i++ )
   {
     printf("Enter value of i");
     scanf("%d", &values[i]);
   }

   for (int i = 0; i < 20; i++) //works out bigggest number
   {
     if(values[i]>big) //compare biggest value with current element
     {
        big=values[i];
```

```
    }
  }

  for (int i = 0; i < 20; i++) //works out smallest number
  {
  if(values[i]<small) //compares smallest value with current element
     {
        small=values[i];
     }
  }

  printf("The biggest number is %d",big ); //prints outs biggest no
  printf("The smallest number is %d ",small); //prints out smalles no
}
```

12. Write a c program to find out second largest element of an unsorted array.

```
#include<stdlib.h>
main()
 {
 int a[100],i,num,n,max;
 printf("enter number of elements to be entered");
 scanf("%d",&n);
 printf("enter numbers");
 for(i=0;i<n;i++)
 scanf("%d",&a[i]);
max=a[0];
num=a[0];
 for(i=0;i<n;i++)
 {
if(a[i]>num)
 {
 if(a[i]>max)
 {num=max;
 max=a[i];}
if(a[i]<max)
 num=a[i];
 }
}
 printf("second largest no. is %d",num);
 system("pause");
}
```

13. Write a c program for delete an element at desired position in an array.

```
#include<stdio.h>
```

```
#include<stdlib.h>
main()
 {
 int a[100],i,n,loc;
printf("enter number of elements to be entered");
 scanf("%d",&n);
 printf("enter numbers");
 for(i=0;i<n;i++)
 scanf("%d",&a[i]);
printf("enter the position at which element has to be deleted");

scanf("%d",&loc);
for(i=loc-1;i<n-1;i++)
 a[i]=a[i+1];
for(i=0;i<n-1;i++)
 printf("%d\n",a[i]);
system("pause");
}
```

14. Write a c program for insert an element at desired position in an array.

```
#include <stdio.h>

void main()
{
    int array[10];
    int i, j, n, m, temp, key, pos;

    printf("Enter how many elements \n");
    scanf("%d", &n);
    printf("Enter the elements \n");
    for (i = 0; i < n; i++)
    {
        scanf("%d", &array[i]);
    }
    printf("Input array elements are \n");
   for (i = 0; i < n; i++)
    {
                printf("%d\n", array[i]);
      }
    for (i = 0; i < n; i++)
    {
        for (j = i + 1; j < n; j++)
        {
            if (array[i] > array[j])
            {
```

```
                temp = array[i];
                array[i] = array[j];
                array[j] = temp;
            }
    }
     }
     printf("Sorted list is \n");
     for (i = 0; i < n; i++)
                {
        printf("%d\n", array[i]);
                }
     printf("Enter the element to be inserted \n");
     scanf("%d", &key);
     for (i = 0; i < n; i++)
     {
    if (key < array[i])
      {
            pos = i;
           break;
        }
     }
     m = n - pos + 1 ;
     for (i = 0; i <= m; i++)
     {
        array[n - i + 2] = array[n - i + 1] ;
     }
     array[pos] = key;
     printf("Final list is \n");
     for (i = 0; i < n + 1; i++)
     {
        printf("%d\n", array[i]);
     }
 }
```

15. **Write a program to swap two integers with and without using temporary variables.**

```
#include<stdio.h>
#include<conio.h>
int main()
{
  int first, second;    // declaring two variables
  printf("Read two values ");
  scanf("%d%d", &first, &second);
first=first+second;
second=first-second;
```

first=first-second;
printf("After swapping the values are: %d, %d ", first, second);
getch();
return 0;
}

16. Write the program to add and subtract two matrices

```
#include <stdio.h>

 void main()
 {
     int array1[10][10], array2[10][10], arraysum[10][10],
     arraydiff[10][10];
     int i, j, m, n, option;

     printf("Enter the order of the matrix array1 and array2 \n");
     scanf("%d %d", &m, &n);
     printf("Enter the elements of matrix array1 \n");
     for (i = 0; i < m; i++)
     {
         for (j = 0; j < n; j++)
         {
             scanf("%d", &array1[i][j]);
         }
     }

     printf("Enter the elements of matrix array2 \n");
     for (i = 0; i < m; i++)
     {
         for (j = 0; j < n; j++)
         {
             scanf("%d", &array2[i][j]);
         }
     }
printf("Enter your option: 1 for Addition and 2 for Subtraction \n");
     scanf("%d", &option);
     switch (option)
     {
     case 1:
         for (i = 0; i < m; i++)
         {
             for (j = 0; j < n; j++)
             {
                 arraysum[i][j] = array1[i][j] + array2[i][j];
             }
         }
        printf("Sum matrix is \n");
         for (i = 0; i < m; i++)
         {
             for (j = 0; j < n; j++)
```

```
                {
                    printf("%3d", arraysum[i][j]) ;
                }
                printf("\n");
            }
                   break;
  case 2:
           for (i = 0; i < m; i++)
           {
          for (j = 0; j < n; j++)
          {
              arraydiff[i][j] = array1[i][j] - array2[i][j];
          }
        }
           printf("Difference matrix is \n");
           for (i = 0; i < m; i++)
            {
          for (j = 0; j < n; j++)
          {
              printf("%3d", arraydiff[i][j]) ;
          }
          printf("\n");
       }

     break;
  }

}
```

17. Program to find the transpose of matrices.

```
#include <stdio.h>

void main()
{
    static int array[10][10];
    int i, j, m, n;

    printf("Enter the order of the matrix \n");
    scanf("%d %d", &m, &n);
    printf("Enter the coefiicients of the matrix\n");
    for (i = 0; i < m; ++i)
    {
        for (j = 0; j < n; ++j)
        {
            scanf("%d", &array[i][j]);
        }
    }
    printf("The given matrix is \n");
    for (i = 0; i < m; ++i)
    {
        for (j = 0; j < n; ++j)
        {
            printf(" %d", array[i][j]);
        }
        printf("\n");
```

```
}
printf("Transpose of matrix is \n");
for (j = 0; j < n; ++j)
{
    for (i = 0; i < m; ++i)
    {
        printf(" %d", array[i][j]);
    }
    printf("\n");
}
```

Exercise

1. What is an array? What is the use of using array? How are they declared in C? what are the rule to be followed while using arrays?

2. Write a C program to multiply two matrices A, B and store the result in C.

3. Write a note on one dimensional and two dimensional arrays

4. Write a C program to read 'n' integer numbers interactively and to print the biggest and smallest of 'n' numbers

5. Given two sets A and B of integers, write a program to read them, determine its UNION and INTERSECTION and print the resultant sets.

6. Write a program to find the intersection of 2 arrays A and B with size m and n respectively

7. Explain initialization and declaration of 2D array

8. Write a c program for insert an element at desired position in an array.

9. Write a c program for delete an element at desired position in an array.

10. Write a c program to find out second largest element of an unsorted array.

11. Write a program to find largest and smallest array in an array

12. Write a program to add all the element of single dimensional array.

13. What is multi dimensional array? How they are declared?

14. Differentiate between i++ and i--?

15. What are the advantages and disadvantages of arrays?

16. Write a program to swap two integers with and without using temporary variables

17. Write the program to add and subtract two matrices

18. Program to find the transpose of matrices.

19. Program to find determinants of matrices.

Quiz:

1. What will happen if in a C program you assign a value to an array element whose subscript exceeds the size of array?

a) The element will be set to 0;
b) The complier may report an error
c) The program may crash if some important data gets overwritten.
d) The array size would appropriately grow.

2. Are the expressions arr and &arr same for an array of 10 integers

a) YES **b) NO**

3. What does the following declaration mean?

int (*ptr) [10];

a) ptr is array of pointers to 10 integers
b) ptr is pointer to an array of 10 integers
c) ptr is an array of 10 integers
d) ptr is an pointers to an array

4. In C, if you pass an array as an argument to a function, what actually gets passed?

a) Value of elements in array b)First elements of the array
c) Base address of the array d) Address of the last elements of the array

5. A pointer to a block of memory is effectively same as an array

a) TRUE b) FALSE

6. Does this mentioning array name gives the base address in all the contexts?

a) YES **b) NO**

7. Is there any difference *int* the following declaration?

```
int fun(int arr[2]);
int fun(int arr[]);
```

a)YES **b)NO**

8. What will be the output of the program ?

```
#include<stdio.h>
int main()
{
   int a[5] = {5, 1, 15, 20, 25};
   int i, j, m;
```

```
i = ++a[1];
j = a[1]++;
m = a[i++];
printf("%d, %d, %d", i, j, m);
return 0;
}
```

a) 2, 1, 15 b) 1, 2, 5 c) 3, 2, 15 d)2, 3, 20

9. The index or subscript value for an array of size n ranges from

a) 1 to n-1 **b) 0 to n-1** c)1 to n d) 0 to n

10. What will be output if you will execute following c code?

```
#include<stdio.h>
void main()
{
   char arr[11]="The African Queen";
    printf("%s",arr);

}
```

a) The African Queen b) Queen c)Queen **d) Null**

11. What will be output if you will execute following c code?

```
#include<stdio.h>
void main()
{
   char arr[7]=”Network”
   printf("%s”, arr);
}
```

a) Network **b)Garbage value** c)Compilation Error d)N

12. The number of elements in array A[3][4] is (JUNE/JULY 2014)

a) 8 **b)12** c)16 d) none of these

13. If A[4] is declaration, then the first and last array index will be (JUNE/JULY 2014)

a) 1,4 **b) 0, 3** c) 3, 0 d) none of these

14. Given A[3][2] = { 1, 2, 3, 4, 5, 6}; The element in 3rd row 2nd col is

a) 3 b) 4 **c) 6** d) 2

STRINGS

In this Lecture we are going to learn the following topics:

- **String variable**
- **Declaring and initializing string variables**
- **Reading and writing strings**

➢ String variable:

A string is an array of characters. Any group of characters defined between double quotation marks is called a constant string.

Example: "Good Morning Everybody"

Character strings are often used to build meaningful and readable programs.

A string variable is any valid C variable name and is always declared as an array.

- **What is a null character?**
  A character constant with an ASCII value of zero is known as the null character and is written as '\0'.

- **What is a character string literal constant? How is it written and stored in the memory?**
  A character string literal constant or just a string literal is a sequence of zero or more characters enclosed within double quotes. For example, "GOD Bless!!" is a string literal constant

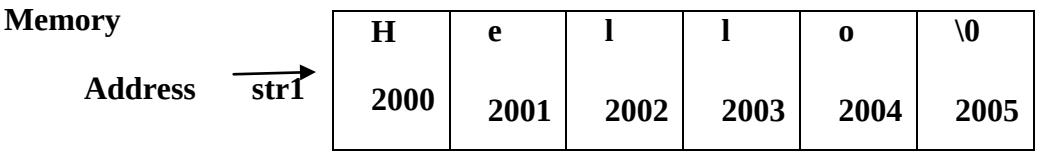
- **What can the maximum number of characters in a character literal constant be?**
  The character constant can be one (e.g. 'A') or two (e.g. '\n') characters long. Hence, the maximum number of characters in a character literal constant can be two.

- **What would be the size of the following arrays:**

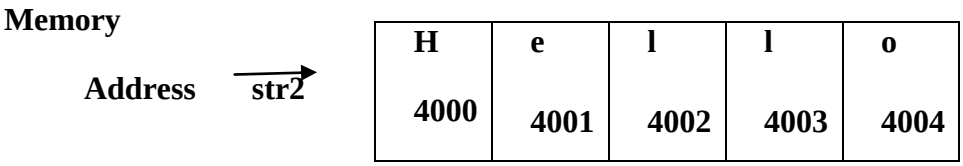
```
char str1[]= "Hello";

char str2[]={'H','e','l','l','o'};
```

The character array str1 is initialized with a character string literal constant "Hello". Since a character string literal constant is terminated by a null character '\0', the contents stored in the character array str1 will be (say array is allocated at 2000):



The character array str2 is initialized with the five initializers in the initialization list. Hence, the contents of str2 will be (say array is allocated at 4000):



Therefore, the size of array str1 is 6 and that of str2 is 5.

- **Declaring and initializing string variables:**

The general form of string variable is

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

The size determines the number of characters in the string-name.

Some examples are:

```
char  state[10];

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.

Character arrays may be initialized when they are declared. C permits a character array to be initialized in either of the following two forms:

```
char state[10]=" KARNATAKA";

char state[10]={'K','A','R','N','A','T','A','K','A'\O'};
```

The reason that state had to be 10 elements long is that the string KARNATAKA contains 10 characters and one element space is provided for the null terminator.

C also permits us to initialize a character array without specifying the number of elements.

For example, the statement: static char string[ ]={'H', 'E', 'L', 'L', 'O' \O};

Defines the array string as a six element array.

➢ Reading and writing strings:

To read a string of characters input function scanf can be used with %s format specification.

Example:

```
char add[20];

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

Note that unlike previous scanf calls, in the case of character arrays, the &(ampersand) is not required before the variable name. 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 the null character.

Here %s ignores the leading characters until next white spaces encountered and end with '\0' (null) characters.

We can input the strings with regular expressions also

Ex: scanf("%s[^\n]", line);

```
#Program to read a series of words using scanf function
        main()
        {
           char text1[50],text2[50],text3[50],text4[50];
           printf("Enter   text:\n");
           scanf("%s  %s", text1,text2);
           scanf("%s", text3);
           scanf("%s", text4);
           printf("\n");
           printf("text1= %s\n text2=%s\n", text1,text2);
           printf("text3=  %s\n text4= %s\n", text3,text4);
}
```

➢ Difference between getc(), getcch(), getche(), getchar()

- **getc ( )**

With this function we Input charater from input stream. it work same as getchar();

- **getchar ( )**
  It will accept a character from keyboard , displays immediately while typing and we need to press Enter key for proceeding.

- **getch ( )**
  It just accepts a keystroke and never displays it and proceeds further. Normally we use it at the end of the main ( ).

- **getche ( )**
  It will accept a character from keyboard , displays it immediately and does not wait for Enter key to be pressed for proceeding

- **gets ( )**
  Both fgets() and gets() read a line terminated with a newline terminator, while gets() reads from stdin and fgets() reads from a specified file. Also, gets() strips the trailing newline from the result.

➢ Writing strings:

The printf function with %s can be used to display an array of characters that is terminated by the null character.

Example:

printf("%s", text);

Can be used to display the entire contents of the array name.

We can also specify the precision with which the array is displayed. For example, the specification

%12.4

indicates that the first four characters are to printed in a field width of 12 columns.

Program to illustrate writing strings using %s format

```
main()
{
        static char state[15]= "MADHYA PRADESH";
        printf("\n \n");
        printf("----------------------------------\n");
        printf("%13s\n", state);
        printf("%5s\n", state);
        printf("%15.6s \n", state);
```

```
        printf("%15.0s\n", state);
        printf("%.3s\n", state);
}
```

Changing string variable:

Any variable of given string can be changed with any other variable

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

str[0]='Y';

printf("Display the given string %s", str);

Output: Yello

STRINGS

In this chapter we are going to learn the following topics:

- **String Handling functions**

➢ String Handling functions:

- **C library supports a large number of string functions. The list given below depicts the string functions**

String Function	Action
strlen(str)	Returns length of the string **str**
strcpy(dest, src)	Copies the source string **src** to destination string **dest**
strncpy(dest, src, n)	Copies at most **n** characters of the source string **src** to destination string **dest**
strcat(str1, str2)	Append string **str2** to string **str1**
strncat(str1, str2, n)	Append first **n** characters of string **str2** to string **str1**
strcmp(str1, str2)	Compares two string **str1** and **str2**
strstr(str1, str2)	Returns a pointer to the first occurrence of **str2** in **str1**, or a null pointer if str2 is not part of str1
strtok(str, delim)	It splits the given string wherever delimiter occurs

➢ Displaying substring of given string:

substr() works with 0, 1 or 2 parameters as both parameters have defaults. The equivalent of the Basic string functions **Left()**, **Mid()** and **Right()** are

String functions	Actions
Left(string,len)- substr(0,len)	It takes specified length of characters from left side of the given string
Mid(string,startpos,len) - substr(startpos,len)	It takes the characters from specified position of the given string based on length of the characters
Right(string,len) - substr(size()-startpos+1, len)	It takes specified length of characters from right side of the given string

- **Left(string,len) - substr(0,len)**

  Ex: Left(str, 7): It displays 7 characters from the left side of the given string.

- **Mid(string,startpos,len) - substr(startpos,len)**

  Ex: Mid(str, 5, 7): It takes the characters from specified position i.e 5 of the given string and displays 7 characters (towards right side of the string).

- **Right(string,len) - substr(size()-startpos+1, len)**

  Ex: Right(str, 7): It displays 7 characters from the right side of the given string.

➢ String Concatenation :strcat() function:

The strcat function joins two strings together. The general form is

strcat(string1,string2);

string1 and string2 are character arrays. When the function strcat is executed. 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.

Example:

Text1= VERY \0

Text2= GOOD\0

Text3= BAD\0

1. strcat(text1,text2);

Text1= VERY GOOD\0

Text2= GOOD\0

2. strcat(text1,text3);
Text1= VERY BAD

Text2= BAD

We must make sure that the size of string1 is large enough to accommodate the final string. Strcat function may also append a string constant to string variable.

For example:

strcat(text1,"GOOD");

C permits nesting of strcat functions. The statement

strcat(strcat(string1,string2),string3);

Is allowed and concatenates all the three strings together. The resultant string is stored in string1.

- **String comparison/strcmp() function:**

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

The general form is :

strcmp(string1,string2);

String1 and string2 may be string variables or string constants.

Examples are:

strcmp(name1,name2);

strcmp(name1, "ABHI");

strcmp("ROM", "RAM");

We have to determine whether the strings are equal, if not which is alphabetically above.

More examples:

strcmp("yello", "hello") : Here strings are not equal and the letter **y** is greater than the letter **h,** therefore the function return the value **> 0.**

strcmp("hello","yello"): Here strings are not equal and the letter **h** is smaller than the letter **y,** therefore the function return the value **< 0.**

strcmp("hi","hi"): Here strings are equal and therefore the function return the value **0.**

strcmpi("Hi","hi"): Here strings are equal because which ignores the case sensitive characters and therefore the function return the value **0.**

- **String copying/strcpy() function:**

The strcpy() function works almost like a string-assignment operator.

The general format is

strcpy(string1,string2);

It copies the contents of string2 to string1. string2 may be a character variable or a string constant.

For example, the statement

strcpy(city , "BANGALORE");

Will assign the string "BANGALORE" to the string variable city.

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.

- **Finding the length of a string/strlen();**

This function counts and returns the number of characters in a string.

The general syntax is **n=strlen(string);**

Where n is an integer variable which receives the value of the length of the string. The argument may be a string constant. The counting ends at the first null character.

Implementing the above functions without using string functions:

- **String concatenation**:
  We cannot assign one string to another directly; we cannot join two strings together by the simple arithmetic addition. The characters from string1 and string2 should be copied into the string3 one after the other. The size of the array string3 should be large enough to hold the total characters.

Program to show concatenation of strings:

```
#include<stdio.h>
 main()
{
    int i,j,k;
    static char first_name={“ATAL”};
    static char sec_name={“RAM”};
    static char  last_name={“KRISHNA”};
    char name[30];

    for(i=0;first_name[i]!=’\0’;i++)
        name[i]=first_name;
    for(i=0;second_name[j]!=’\0’; j++)
        name[i+j+1]=sec_name[j];
         name[i+j+1] =’ ‘;
    for(k=0;last_name[k]!=’\0’;k++)
         name[i+j+k+2]=last_name[k];
         name[i+j+k+2]=’\0’;
          printf(“\n \n”);
           printf(“%s \n”, name);

}
```

Output

ATAL RAM KRISHNA

- **String comparison:**
  Comparison of two strings cannot be compared directly. It is therefore necessary to compare the strings to be tested, character by character. The comparison is done until there is a mismatch or one of the strings terminates into a null character.

  The following segment of a program illustrates this,

Program to show comparison between two strings:

```
#include<stdio.h>
        main ()
 {
   char str1[10], str2[10];
   int i;
      printf("Read first string");
      scanf("%s", str1);
      printf("Read second string");
      scanf("%s", str2);
      i=0;
      while(str1[i]==str2[i] && str1[i]!='\0' && str2[i]!='\0')
        {
              i=i+1;
        if(str1[i]=='\0' && str2[i]=='\0')
              printf("strings are equal\n");
          else
              printf("strings are not equal\n");
        }
 }
```

- **String copying:**

strcpy(dest, src): If ***'dest'*** and ***'src'*** contains same string, then if we want to copy the string from source 'src' to the destination 'dest' while coping it overlaps the contents then the results are unpredictable or it displays the garbage value.

Program to show copying of two strings:

```
#include<stdio.h>
main()
{
    char string1[80],string2[80];
    int j;
    printf("Enter a string\n");
    printf("?");
    scanf("%s", string2);
     for(j=0;string2[i]!='\0';j++)
      string1[j]=string2[j];
      string1[j]='\0';
      printf("\n");
      printf("%s\n",string1);
      printf("Number of characters=%d\n", j);
}
```

Program to find the length of a string:

```
#include<stdio.h>
main()
    {
        char line[80],character
        int c=0,i;
        printf("Enter the text\n");
        for(i=0;line[i];!='\0';i++)
        {
            character=getchar();
            line[i]=character;
            c++;
        }
       printf("The length of the string \n", c);
    }
```

➢ Arithmetic operations on characters:

We can manipulate characters the same way we do with numbers. 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.

To write a character in its integer representation, we may write it as an integer.

For example:

```
y='a';
printf("%d\n", y);
```

will display the number 97 on the screen.

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

For example:

```
y='z'-1;
```

Is a valid statement. In ASCII , the value of 'z' is 122 and therefore , the statement will assign the value 121 to the variable Y.

We may also use character constants in relational expressions.

For example:

```
ch>='a' && ch<='z'
```

Would test whether the character contained in the variable ch is an lower-case letter.

We can convert a character digit to its equivalent integer value using the following relationship:

```
y=character –'0';
```

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

For example: Let us assume that the character contains the digit '7', then,

```
y=ASCII value of '7'-ASCII value of '0'
= 55-48
=7
```

C library has a function that converts a string of digits into their integer values. The function takes the form

y=atoi(string);

y is an integer variable and string is a character array containing a string or digits

For example:

num="1974"

year=atoi(num);

Num is a string variable which is assigned the string constant "1974". The function atoi converts the string "1974" to its numeric equivalent 1974 and assigns it to the integer variable year.

- **What is the difference between a string copy (*strcpy*) and a memory copy (*memcpy*)? When should each be used?**

The *strcpy()* function is designed to work exclusively with strings. It copies each byte of the source string to the destination string and stops when the terminating null character (*\0*) has been moved. On the other hand, the *memcpy()* function is designed to work with any type of data.

Because not all data ends with a null character, you must provide the *memcpy()* function with the number of bytes you want to copy from the source to the destination.

- **How can I convert a number to a string?**

The standard C library provides several functions for converting numbers of all formats (integers, longs, floats, and so on) to strings and vice versa. One of these functions, *itoa()*, is used here to illustrate how an integer is converted to a string:

```
#include <stdio.h>
#include <stdlib.h>
void main(void);
void main(void)
{
    int num = 100;
    char str[25];
    itoa(num, str, 10);
    printf("The number 'num' is %d and the string 'str' is %s.\n", num, str);
}
```

Explanation: Here that the *itoa()* function takes three arguments: the first argument is the number you want to convert to the string, the second is the destination string to put the converted number into, and the third is the base, or radix, to be used when converting the number. But this example uses the common base 10 to convert the number to the string.

The following functions can be used to convert integers to strings:

Function name		Purpose
itoa( )	-	Converts an integer value to string
Itoa( )	-	Converts a long integer value to string
utoa( )	-	Converts an unsigned long integer value to string
strtoul()	-	Converts string to unsigned long integer

Note: the *itoa()*, *ltoa()*, and *ultoa()* functions are not ANSI compatible.

ASCII values for the alphabets: A to Z : 65 to 90 and

a to z : 97 to 122

STRINGS

In this chapter we are going to learn the following topics:

- **Programming examples**
- **Exercise**
- **Quiz**

Programming examples:

#Program to sort strings in alphabetical order:

```
#define ITEMS 10
#define MAX   25
main()
{
        char str [ITEMS][MAX], dum[MAX];
        int i=0;j=0;
        printf("Enter names of %d items \n", ITEMS);
        while(i<ITEMS)
        scanf("%s", str[i++]);

        for(i=1;i<ITEMS;i++)
    {
        for(j=1;j<=ITEMS-i;j++)
         {
           if(strcmp(string[j-1],string[j])>0)
                strcpy(dummy,string[j-1]);
                strcpy(str[j-1],str[j]);
                strcpy(str[j],dummy);
        }
    }
        for(i=0;i<ITEMS;i++)
          printf("%s", str[i]);
  }
```

#Program to show string handling functions:

```
#include<string.h>
main()
{
   char s1[20],s2[20],s3[20];
    int y,len1,len2,len3;
    printf("\n Enter two string constants\n");
    printf("?");
    scanf("%s %s", s1,s2);
```

```
        x=strcmp(s1,s2);
         If(y!=0)
            {
              printf("\n\n Strings are not equal\n");
               strcat(s1,s2);
            }
      else
            printf("\n\n Strings are equal\n");
            strcpy(s3,s1);
            len1=strlen(s1);
            len2=strlen(s2);
            len3=strlen(s3);
            printf("\n s1= %s length= %d character \n",s1,len1);
            printf("\n s1= %s length= %d character \n",s2,len2);
             printf("\n s1= %s length= %d character \n",s3,len3);
            }
```

#Program to convert lowercase characters in to upper case characters:

```
          #include<stdio.h>
           main()
           {
                    char text[85];
                    int i=0;
                    printf("Enter a line of text in lowercase:\t");
                    scanf("%[^\n]",text);
                    printf("%s",text);
                    printf("\n Converted to uppercase text is :\t");
                    while(text[i]!='0')
                   {
                      printf("%c", toupper(text[i]));
                       i++;
                    printf("\n");
            }
```

#Write a 'C' program to accept a string from user and delete all the vowels from that string & display the result.

```
 #include<stdio.h>
#include<conio.h>
void main()
{
int i=0,flag=0;
char s[50];
clrscr();
printf("\n enter the string:-");
gets(s);
printf(" the string is =%s",s);
printf("\n");
while(s[i]!=NULL)
{
if(s[i]=='a'||s[i]=='e'||s[i]=='i'||s[i]=='o'||s[i]=='u'||s[i]=='A' || s[i]=='E' || s[i]=='I'|| s[i]=='O' ||
s[i]=='U')
{
flag=1;
}
else
{
printf("%c",s[i]);
}
i++;
}
getch();
}
```

#Write a C program to delete all consonants from a sentence.

```
#include <stdio.h>
 #include <string.h>

 int main() {
     int i = 0, j = 0, k;
     char string[256], result[256];

     /* vowels in lowercase and uppercase */
     char vowel[10] = {'a', 'e', 'i', 'o', 'u',
                       'A', 'E', 'I', 'O', 'U'};

     /* get the input string from the user */
     printf("Enter your input string:");
     fgets(string, 256, stdin);
     string[strlen(string) - 1] = '\0';

     /* copy consonants alone to result array */
```

```
        while (string[i] != '\0') {
            for (k = 0; k < 10; k++) {
                if (vowel[k] == string[i]) {
                    result[j++] = string[i];
                    break;
                }
            }

            i++;
        }

        result[j] = '\0';

        /* copy the contents of result to string */
        strcpy(string, result);

        /* print the result */
        printf("After deleting all consonants: %s\n", string);
        return 0;
    }
```

#Program to convert uppercase to lowercase and vice versa

```
#include <iostream>
#include <string>
using namespace std;
int main(){
    string s;
    cout<<"enter the string :"<<endl;
    cin>>s;
    for (int i=0;i<s.length();i++){
      if ('a'<=s[i] && s[i]<='z'){
        s[i]=char(((int)s[i])-32);
      }

      else if('A'<=s[i] && s[i]<='Z'){
        s[i]=char(((int)s[i])+32);
      }
     }

    cout<<"modified string is  : "<<s<<endl;
    return 0;
}
```

5.10 Exercise

1. What is string? How strings are represented?
2. What is null string? How null strings are represented**?**
3. What is a character string literal constant? How is it written and stored in the memory?
4. What can the maximum number of characters in a character literal constant be?
5. What is pointer? Explain with example.
6. What are the string handling function?
7. Write a program to show pointers in one-dimensional arrays
8. Write a program to convert lowercase characters in to upper case characters
9. Writ a program to sort strings in alphabetical order.
10. Difference between strcmp() and strcmpi().
11. What is difference between getc() and gets()?
12. What is difference between getch() and getche()?
13. Write a program to find a substring in a given strings.
14. Write a c program to delete the all consonants from given string.
15. Program to convert uppercase to lowercase and vice versa
16. Program to find the string length without using built-in function.
17. Write Difference between getc(), getcch(), getche(), getchar()
18. Mention some built-in functions of string.h header file.
19. What is isalpha() and tolower() functions?
20. What is strlen(), strcpy(), strcat() and strcmp() function works?

Quiz

1. Which of the following function sets first n character of the string to a given character?
   a) strinit() **b) strnset()** c) strset() d) strcset()2

2. If the two strings are identical, then strcmp() function return
   a) -1 b) 1 **c) 0** d) yes

3. How will you print \n on the screen?
   a) printf("\n"); b) echo "\\n"; c) printf('\n'); **d) printf("\\n");**

4. The library function used to find the last occurrence of a character in a string is
   a) strnstr() b) laststr() **c) strrchr()** d) strstr()

5. Which of the following function is used to find the first occurrences of a given string in another string?
   a) strchr() b) strrchr() **c) strstr()** d) strnset()

6. Which of the following is more appropriate for reading in multi-word string?
   a) printf(); b) scanf(); **c) gets();** d) puts();

7. What is the output of this C code?

```
#include <stdio.h>
void foo(int*);
int main()
{
   int i = 10;
   foo((&i)++);
}

  void foo(int *p)

  {
   printf("%d\n", *p);
  }
```

a) 10 b) some garbage value c) compiler error d)segmentation fault

8. The return-type used in String operations are. *

a)void only **b) void & (char *) only** c) void & int only

d) void, int & (char *) only

9. C-strings cannot be concatenated by + operator

a) True **v) False**

10. The end of the string is marked with the help of _____ character.

a)return(enter) b) **null** **c)** tab d) semicolon

.