1 What is an array? Explain with Example. What are the advantages of using an array?

- An array is a fixed-size sequenced collection of elements of the same data type.
- An array is derived data type.
- The individual element of an array is referred by their index or subscript value.
- The subscript for an array always begins with 0.

Syntax : data_type array_name [size];
Example : int marks [5];

- The *data_type* specifies the type of the elements that can be stored in an array, like int, float or char.
- The *size* indicates the maximum number of elements that can be stored inside the array.
- In the example, data type of an array is int and maximum elements that can be stored in an array are 5.

Advantages:

- You can use one name to store many values with different indexes.
- An array is very useful when you are working with sequences of the same kind of data.
- An array makes program easier to read, write and debug.

Disadvantages:

- We must know size of an array in advance before value stored in array.
- It is static structure, the memory which is allocated to array cannot be increased or decreased.
- Array's size is fixed, so if we allocate more memory than requirement then it will be wastage of memory.
- The elements of array are stored in consecutive memory locations.
- Insertions and deletions are very difficult and time consuming.

Example:

```
#include<stdio.h>
void main()
{
     int a[5] = {5,12,20,54,68}, i ;
     for(i=0; i<5; i++)
     {
          printf(“%d”, a[i]);
     }
}
```

Types of an array:

1) **Single dimensional array**
2) **Two dimensional array**
3) **Multi-dimensional array**

2 Explain initialization and working of single and multi-dimensional array with example.

Single Dimensional Array:

- An array using only one subscript to represent the list of elements is called single dimensional array.

Syntax : data_type array_name [size];

Example : int marks [5];

- An individual array element can be used anywhere like a normal variable with a statement such as
  g = marks [60];
- More generally if *i* is declared to be an integer variable, then the statement **g=marks[i];** will take the value contained at **i**th position in an array and assigns it to **g**.
- We can store value into array element by specifying the array element on the left hand side of the equals sign like **marks[60]=95;** The value 95 is stored at **60th** position in an array.
- The ability to represent a collection of related data items by a single array enables us to develop concise and efficient programs.
- **For example** we can very easily sequence through the elements in the array by varying the value of the variable that is used as a subscript into the array.

```
for(i=0; i<66; i++);
        sum = sum + marks[i];
```

- Above for loop will sequence through the first 66 elements of the marks array (elements 0 to 65) and will add the values of each marks into sum. When for loop is finished, the variable sum will then contain the total of first 66 values of the marks.
- The declaration **int values[5];** would reserve enough space for an array called values that could hold up to 5 integers. Refer to the below given picture to conceptualize the reserved storage space.

values[0]	
values[1]	
values[2]	
values[3]	
values[4]	

Initialization of Single Dimensional array:

- The general form of initialization of array is:
  data_type array_name [size] = {list of values} ;
- There are three ways to initialize single dimensional array,

1. int number [3] = {1, 5, 2} ;
   will initialize **0th** element of an array to **1**, **1st** element to **5** and **2nd** element to **2**.

2. int number [5] = {1, 7} ;
   will initialize **0th** element of an array to **1**, **1st** element to **7** and rest all elements will be initialized to **0**.

3. int number[ ] = {1, 5, 6} ;
   first of all array size will be fixed to **3** then it will initialize **0th** element to **1**, **1st** element to **5** and **2nd** element to **6.**

Two dimensional arrays:

- Two dimensional arrays are also called table or matrix.
- Two dimensional arrays have two subscripts.
- First subscript denotes the number of rows and second subscript denotes the number of columns.

Syntax : data_type array_name [row_size] [column_size];

Example : int marks [10][20];

- Here m is declared as a matrix having 10 rows (numbered from 0 to 9) and 20 columns (numbered 0 through 19). The first element of the matrix is m[0][0] and the last row last column is m[9][19]
- A two dimensional array marks[4][3] is shown below. The first element is given by marks[0][0] contains 35.5 & second element is marks[0][1] and contains 40.5 and so on.

marks [0][0] 35.5	marks [0][1] 40.5	marks [0][2] 45.5
marks [1][0] 66.5	marks [1][1] 55.5	marks [1][2] 60.5
marks [2][0] 85.5	marks [2][1] 78.5	marks [2][2] 65.3
marks [3][0] 25.6	marks [3][1] 35.2	marks [3][2] 76.2

Initialization of two dimensional array:

1. int table [2][3] = {1,2,3,4,5,6};
   will initialize **1st** row **1st** column element to **1**, **1st** row **2nd** column to **2**, **1st** row **3rd** column to **3**, **2nd** row **3rd** column to **6** and so on.
2. int table [2][3] = {{1,2,3},{4,5,6}};
   here, **1st** group is for **1st** row and **2nd** group is for **2nd** row. So **1st** row **1st** column element is **1**, **2nd** row **1st** column element is **4**, **2nd** row **3rd** column element is **6** so on.
3. int table [2][3] = {{1,2},{4}}
   initializes as above but missing elements will be initialized by 0.

3. Insertion Operation:

- Insert means to add new element into an array at specified Position. Considering C array having Maximum Size 10 having currently n=5 elements, to insert an element at index 1, first all the elements from position 1 to n-1 are shifted down by one position to make space at index 1 then new value is to be copied at index 1.

10
17
12
8
24

Before Insert Element

10
50
17
12
8
24

After Insert Element

Example:

```
#include<stdio.h>
void main( )
{
int a[100],k,i,n,val,pos;
printf("Enter the size of the array:");
scanf("%d",&n);
printf("\n Enter array elements:");
for(i=0; i<n; i++)
{
scanf("%d", &a[i]);
}
printf("\nEnter Position to insert : ");
scanf("%d",&pos);
printf("\n Enter Value:");
scanf("%d",&val);
for(k=n-1;k>=pos-1;k--)
{
   a[k+1] = a[k];
}
a[pos-1]= val;
printf("\n After insertion array is :\n");
for(i=0; i<=n; i++)
{
printf("\n%d", a[i]);
}
getch();
}
```

4. Deletion Operation:

- Delete operation removes an element from the specified index. This requires all the elements from next position to last position to be shifted up.
- After completing delete operation, size of the array is reduced by 1 as it deletes one element from an array.

Example:

```
#include<stdio.h>
void main()
{
int array[100], position, c, n;
printf("Enter number of elements in array:");
scanf("%d", &n);
printf("Enter %d elements:\n", n);
for ( c = 0 ; c < n ; c++ )
scanf("%d", &array[c]);
printf("Enter the location where you wish to delete
element\n");
scanf("%d", &position);
if ( position >= n+1 )
printf("Deletion not possible.\n");
else
{
for ( c = position - 1 ; c < n - 1 ; c++ )
{
array[c] = array[c+1];
}
printf(" After Deletion New Array is:\n");
for( c = 0 ; c < n - 1 ; c++ )
printf("%d\n", array[c]);
 }
getch();
}
```

5. Sorting Operation:

- The sorting array means to arrange elements in an ascending or descending order this is very important operation used in the database.
- The sorting is performed by comparing and exchanging the elements of an array if they are not in order.

Example:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int array[100], i, j, n, swap;
printf("Enter Number of Elements:\n");
scanf("%d",&n);
printf("Enter Element Values:");
for(i=0; i<n; i++)
{
scanf("%d", &array[i]);
}
for(i=0; i<n-1; i++)
{
for(j=i+1;j<n; j++)
{
if (array[i] > array[j])
{
swap = array[i];
array[i] = array[j];
array[j] = swap;
       }
  }
}
printf("Sorted Array:");
for(i=0;i<n;i++)
  printf("\n%d",array[i]);
getch();
}
```

6. Searching Operation:

- It is an operation used to search an element from the array.

  Example:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int array[100], search, c, n;
printf("Enter the number of elements in array:\n");
scanf("%d",&n);
printf("Enter %d integers:\n", n);
for (c = 0; c < n; c++)
{
scanf("%d", &array[c]);
}
printf("Enter the number to search\n");
scanf("%d", &search);
for (c = 0; c < n; c++)
{
if (array[c] == search)
{
```

```
printf("%d is present at location %d.\n", search, c+1);
break;
}
}
if (c == n)
printf("%d is not present in array.\n", search);
getch();
}
```

7. Merging Operation:

- The merging operation joins two arrays one after another.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a1[5],a2[3],a[8];
int i,j;
printf("Enter First array:");
for(i=0;i<5;i++)
scanf("%d",&a1[i]);
printf("Enter Second array:");
for(i=0;i<3;i++)
scanf("%d",&a2[i]);
for(i=0,j=0;i<5;i++,j++)
          a[j]=a1[i];
for(i=0;i<3;i++,j++)
          a[j]=a2[i];

printf("Merged array:");
for(i=0;i<8;i++)
     printf("\n%d",a[i]);
getch();
}
```

8. Matrix addition Program using 2-Dimensional array:

```
#include<stdio.h>
#include<conio.h>
void main()
{
     int mat_1[3][3],mat_2[3][3],i,j,k,sum=0;
     printf("\nEnter 1st Matrix here");
     for(i=0;i<3;i++)
     {
          for(j=0;j<3;j++)
          {
               scanf("%d",&mat_1[i][j]);
          }
     }
     printf("\nEnter 2nd Matrix here");
     for(i=0;i<3;i++)
     {
          for(j=0;j<3;j++)
          {
               scanf("%d",&mat_2[i][j]);
          }
     }
    printf("Sum of entered matrices:-\n");
    for ( i = 0 ; i < 3 ; i++ )
    {
      for ( j = 0 ; j < 3 ; j++ )
        printf("%d\t", mat_1[i][j]+mat_2[i][j]);
      printf("\n");
    }
     getch();
}
```

9. How to declare and initialize a string. Explain with example.

- The general form of declaration of a string variable is:
  Syntax: char string_name[size];
- The size determines the number of characters in the string_name. Some examples are:
  Example: char city[10];
  char name[30];
- When the compiler assigns a character string to a character array, it automatically supports a null character ('\0') at the end of the string.
- Therefore, size should be equal to the maximum number of characters in the string plus one.
- Character arrays can be initialized in the following forms:
  Example: char str[8] = "COMPUTER";
  char str[8] = {'C', 'O', 'M', 'P', 'T', 'E', 'R', '\0'};
- C also permits us to initialize a character array without specifying the number of elements. Here, size of the array will be determined automatically, based on the number of initialized.
  Example: char str[] = {'G', 'O', 'O', 'D', '\0'};
  char str[] = "GOOD";
- Defines the array str as a five element array.
- We can also declare the size much larger than the string size in the initialize. For
  Example: char str[10] = 'GOOD'; is permitted.

10. What is a string? What are the operations that can be performed on string?

- String is a sequence of characters enclose in double quotes.
- String is collection of character array elements.
- Normally string is useful for storing data like name, address, city etc.
- ASCII code is normally used to represent string in memory.
- In ‘C’ each string is terminated by a special character called a NULL character.
- In ‘C’ 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.

  Example:

C	O	M	P	U	T	E	R	\0

- Here, the string is stored is “COMPUTER”, which is having only 8 characters, but actually 9 characters are stored because of NULL character at the end.

Operations on String:

1) String Length
2) String Copy
3) Reverse a String
4) Compare two Strings
5) Convert String to upper case
6) Convert String to lower case
7) String Concatenation

12. How to read/write strings from terminal?

- The input function scanf() can be used with **%s** format specification to **read** a string of characters.

  Example: char address[10];
  scanf (“%s”, address);
- Note: When using **scanf()** function for string, the scanf() does not use **‘&’** symbol for scanning a string.
- We can also use the function **gets()** for reading a string.
- The array should be written within brackets.

  Example: char name[20];
  gets (name);
- The string can be printed using **printf()** function with **%s** format specifier, or by using **puts()** function as shown below:

  Example: char name[] = “Tom”;
  printf(“%s”, name);
  puts(name);

13 Explain difference between scanf() and gets(), printf() and puts()

scanf()	gets()
✓ It is used to read **all types** of data.	✓ It is used to read only **string** data.
✓ It is terminated by using white space.	✓ It is terminated by enter key or at end of line.
✓ It cannot read white space between two words of a string.	✓ It is used to read complete string with white spaces.
✓ It requires format specifier to read formatted data.	✓ It can read single string at a time, It does not require format specifier.
✓ Syntax: scanf(“list of format specifier”, lis of addresses of variable);	✓ Syntax: gets(String_variable);

✓ **Example:**
int a,b;
scanf (“%d%d”,&a,&b);
	✓ **Example:**
	char ch[10];
	gets(ch);

printf()	puts()
✓ It is used to display all types of data and messages.	✓ It is used to display only string data and messages.
✓ It requires format specifier to display formatted data.	✓ It does not requires format specifier to display string.
✓ It can display multiple data at a time by multiple format specifier in one printf( ).	✓ It is used to display only one string at a time.
✓ Syntax: printf(“list of format specifier or message”, list of variables);	✓ Syntax: puts(variable);

✓ **Example:**
int a,b;
printf (“%d%d”,a,b);
	✓ **Example:**
	char ch[]=”Hello”;
	puts(ch);

14. Explain various string handling operations available in ‘C’ with example.

- C has several inbuilt functions to operate on string. These functions are known as string handling functions.
- **Example:**
  char s1[20]= ”Computer”, s2[8]= ”Komputer”;

Function	**Meaning**
l=strlen(str1)	Returns length of the string. l=strlen(str1); It Returns **8**
strcmp(str1,str2)	Compares two strings. It returns negative value if str1<str2, or positive value if str1>str2 and zero if str1=str2 printf("%d",strcmp(str1,str2)); **OUTPUT : -1**
strcpy(str1,str2)	Copies 2nd String in to the 1st String strcpy(str1,str2); copies the string str2 in to the sting str1, **so str1 is now "Komputer"** str2 remains unchaged
strcat(str1,str2)	Appends 2nd String at the end of 1st String strcat(str1,str2); a copy of string str2 is appended at the end of string str1. **Now str1 becomes "ComputerKomputer"**
strchr(s1,c);	Returns a pointer to the first occurrence of a given character in the string str1. printf("%s", strchr(str1,'e'); **OUTPUT : er**
strstr(str1,str2)	Returns a pointer to the first occurrence of a given string str2 in string str1. printf("%s",strstr(str1,"ter"); **OUTPUT: ter**
strrev(str1);	Reverse the given string. strrev(str1); **makes string str1 to "retupmoC"**
strlwr(str1);	Converts string str1 to lower case printf("%s",strlwr(str1)); **OUTPUT : computer**
strupr(str1);	Converts string str1 to upper case printf("%s",strupr(str1)); **OUTPUT : COMPUTER**

strncpy(str1,str2,n)	Copies first n characters of string str2 to string str1
	str1=""; str2="ComputerProgramming";
	strncpy(str1,str2,8);
	printf("%s",str1); **OUTPUT: Computer**

strncat(str1,str2,n)	Appends first n characters of string str2 at the end of the string str1. strncat(str1,str2,2); printf("%s",str1); **OUTPUT: ComputerKo**
strncmp(str1,str2,n)	Compares first n characters of string str1 and str2 and returns similar result as strcmp() function. str1="Computer";str2="Komputer"; printf("%d",strncmp(str1,str2,5)); **OUTPUT: -13**
strrchr(str1,c)	Returns the last occurance of a given character in a string str1. str1="ComputerProgramming"; printf("%s",strrchr(str1,'m'); **OUTPUT: ming**