As discussed earlier while reading about accessing array elements, traversing the array element accessing each element of the array for a specific purpose.

If A is an array of homogeneous data elements, then traversing the data elements can include printing element, counting the total number of elements, or performing any process on these elements. As is a linear data structure because all its elements form a sequence, traversing its elements is very straightforward. The algorithm for array traversal is given in Fig. 6.11.

```
Step 1: [Initialization] Set I = lower_bound
Step 2: Repeat steps 3 to 4 while I <= upper_bound
Step 3: Apply Process to A[I]
Step 4: Set I = I + 1
        [End of Loop]
Step 5: Exit
```

Figure 6.11 Alogrithm for array traversal

In Step 1, we initialize index to the `lower bound` of the array. In Step 2, a `while` loop is executed. Steps 3 and 4 form a part of the loop. Step 3 processes the individual array element as specified by the array name and index vale. Step 4 increments the index value so that the next array element could be processed. The loop in Step 2 is executed until all the elements in the array are processed. In other words, until `I` is less than or equal to the `upper bound` of the array.

Example 6.4 Assume that there is an array `Marks[]`, such that the index of the array specifies the roll number of the student and the value of a particular element denotes the marks obtained by the student. For example, if it is given `Marks[4] = 78`, then the student whose roll number is 4 has obtained 78 marks in the examination. Now, write an algorithm to

- Find the total number of students who have secured 80 or more marks
- Print the roll number and marks of all the students who have got distinction

```
   Step 1: [Initialization] Set I = I + 1
   Step 2: Repeat for I = lower_bound to upper_bound
           If Marks[I] >= 80, then : Set Count = Count + 1
           [End of Loop]
   Step 3: Exit
b) Step 1: [Initialization] Set I = I + 1
   Step 2: Repeat for I = lower_bound to upper_bound
                If Marks[I] >= 75,  Write: I , Marks[I]
               [End of Loop]
   Step 3: Exit
```

Program 6.1 Write a program to read and display n numbers using an array.

```
using namespace std;
#include<iostream>
int main()
{   int i = 0, n, arr[20];
    cout<< "\n Enter the number of elements : ";
    cin>>n;
    for(i = 0;i < n;i++)
```

```
    {   cout<<"\n Arr["<<i<<"] = ";
        cin>>arr[i];
    }
    cout<<"\n The array elements are \n";
    for(i = 0;i < n;i++)
    cout<<"\t Arr["<<i<<"] = "<<arr[i];
}
```

OUTPUT

```
Enter the number of elements : 5
1 2 3 4 5
The array elements are
Arr[0] = 1 Arr[1] = 2 Arr[2] = 3 Arr[3] = 4 Arr[4] = 5
```

Program 6.2 Write a program to read and display n random numbers using an array.

```
using namespace std;
#include<iostream>
#include<stdlib.h>
#define MAX 10
main()
{   int arr[MAX], i,RandNo;
    for(i = 0;i < MAX;i++)
    {   // Scale the random number in the range 0 to MAX-1
        RandNo = rand() % MAX;
    arr[i] = RandNo;
    }
    cout<<"\n The contents of the array are :- ";
    for(i = 0;i < MAX;i++)
        cout<<"\t"<<arr[i];
}
```

OUTPUT

```
The contents of the array are :-7  0  1  2  4  6  5  4  8  3
```

Program 6.3 Write a program to find largest of n numbers using arrays.

```
using namespace std;
#include<iostream>
int main()
{   int i, n, arr[20], large = 0;
    cout<<"\n Enter the number of elements in the array : ";
    cin>>n;
    cout<<"\n Enter the elements :\n";
    for(i = 0;i < n;i++)
        cin>>arr[i];
    large = arr[0];
    for(i = 1;i < n;i++)
    {   if(arr[i] > large)
        large = arr[i];
    }
    cout<<"\n The largest number is : "<<large;
}
```

OUTPUT

```
Enter the number of elements : 5
1 2 3 4 5
The largest number is : 5
```

Program 6.4 Write a program to interchange the largest and the smallest number in the array.

```
using namespace std;
#include<iostream>
int main()
{   int i, n, arr[20], temp;
    int small, small_pos = 0, large, large_pos = 0;
    cout<<"\n Enter the number of elements in the array : ";
    cin>>n;
    cout<<"\n Enter the elements :\n";
    for(i = 0;i < n;i++)
        cin>>arr[i];
small = large = arr[0];
for(i = 0;i < n;i++)
{   if(arr[i] < small)
    {   small = arr[i];
        small_pos = i;
    }
    if(arr[i] > large)
    {   large = arr[i];
    large_pos = i;
    }
}
cout<<"\n The smallest of these numbers is : "<<small;
cout<<"\n The position of the smallest number in the array is : "<<small_pos;
cout<<"\n The largest of these numbers is : "<<large;
cout<<"\n The position of the largest number in the array is : "<<large_pos;
temp = arr[large_pos];
arr[large_pos] =arr[small_pos];
    arr[small_pos] = temp;
    cout<<"\n The new array is : ";
    for(i = 0;i < n;i++)
        cout<<" \n arr["<<i<<"] = "<<arr[i];
}
```

OUTPUT

```
Enter the number of elements : 5
1 2 3 4 5
The smallest number is : 1
The position of the smallest number in the array is : 0
The largest number is : 5
The position of the largest number in the array is : 4
The new array is
5 4 3 2 1
```

Program 6.5 Write a program to find the second largest number using an array of n numbers.

```
using namespace std;
#include<iostream>
int main()
{   int i, n, arr[20], large, second_large;
    cout<<"\n Enter the number of elements in the array : ";
    cin>>n;
    cout<<"\n Enter the elements :\n";
    for(i = 0;i < n;i++)
        cin>>arr[i];
    large = second_large = arr[0];
```

```
for(i = 0;i < n;i++)
{   if(arr[i]>large)
        large = arr[i];

}
for(i = 0;i < n;i++)
{   if(arr[i] != large)
    {
        if(arr[i]>second_large)
            second_large = arr[i];
    }
}
cout<<"\n The numbers you entered are : ";
for(i = 0;i < n;i++)
    cout<<arr[i];
    cout<<"\n The largest of these numbers is : "<<large;
    cout<<"\n The second largest of these numbers is : "<<second_large;
}
```

OUTPUT

```
Enter the number of elements : 5
1 2 3 4 5
The numbers you entered are :
1 2 3 4 5
The largest of these numbers is : 5
The second largest of these numbers is : 4
```

Program 6.6 Write a program to enter n number of digits and then form a number using these digits.

```
using namespace std;
#include<iostream>
#include<math.h>
int main()
{   int number = 0, digit[10], numofdigits,i;
    cout<<"\n Enter the number of digits : ";
    cin>>numofdigits;
    for(i = 0;i < numofdigits;i++)
    {   cout<<"\n Enter the "<<i<<" th digit : "<<i;
        cin>>digit[i];
    }
    i = 0;
    while(i < numofdigits)
    {   number = number + digit[i] * pow(10,i);
        i++;
    }
    cout<<"\n The number is : "<<number;
}
```

OUTPUT

```
Enter the number of digits : 3
Enter the 0th digit : 3
Enter the 1th digit : 4
Enter the 2th digit : 5
The number is : 543
```

Program 6.7 Write a program to find whether the array of integers contain a duplicate number.

```
using namespace std;
#include<iostream>
int main()
```

```
{   int array1[10], i, n, j, flag = 0;
    cout<<"\n Enter the number of elements in the array : ";
    cin>>n;
    cout<<"\n Enter the elements :\n";
    for(i = 0;i <n ;i++)
        cin>>array1[i];
    for(i = 0;i < n;i++)
    {   for(j = i + 1;j < n;j++)
    {   if(array1[i] == array1[j] && i!=j)
        { flag = 1;
          cout<<"\n Duplicate number "<<array1[i]<<" found at location "<<i<<" and "<<j;
        }
        }
        if(flag = 0)
        cout<<"\n No Duplicate";
}
```

OUTPUT

```
Enter the number of elements : 5
1 2 3 4 5
No Duplicate
```

Program 6.8 Write a program to read marks of 50 students in the range of 0–100. Then make 10 groups such as 0-10, 10-20, 10-20, and so on. Count the number of values that falls in each group. And display the result.

```
using namespace std;
#include<iostream>
main()
{   int marks[50], i;
    int group[10] = {0};
    cout<<"\n Enter the marks of 50 students : \n";
    for(i = 0;i < 50;i++)
    {   cout<<"\n MARKS["<<i<<"] = ";
        cin>>marks[i];
        ++group[(int)(marks[i])/10];
    }
    cout<<"\n\n **************************";
    cout<<"\n GROUP \t\t FRQUENCY";
    for(i=0;i<10;i++)
    cout<<"\n "<<i<<" \t\t "<<group[i];
}
```

SAMPLE OUTPUT

```
**************************
GROUP     FRQUENCY
0          4
1          2
2          6
3          5
4          10
5          22
6          18
7          9
8          5
9          4
10         0
```

Program 6.9 Modify the previous program to display frequency histograms of each group.

```
using namespace std;
#include<iostream>
main()
{   int marks[50], i;
    int group[10] = {0};
    cout<<"\n Enter the marks of 50 students : \n";
    for(i = 0;i < 50;i++)
    {   cout<<"\n MARKS["<<i<<"] = ";
    cin>>marks[i];
    ++group[(int)(marks[i])/10];
    }
    cout<<"\n\n ***************************";
    cout<<"\n GROUP \t\t FREQUENCY";
    for(i = 0;i < 10;i++)
        cout<<"\n "<<i<<" \t\t "<<group[i];
    cout<<"\n\n FREQUENCY HISTOGRAM";
    for(i = 0;i < 10;index++)
    cout<<"\n GROUP "<<i<<" | ";
    for(i = 0;i < group[i];i++)
    cout<<" *";
}
```

SAMPLE OUTPUT

```
GROUP    FREQUENCY
0    ****
1    **
2    ******
3    *****
```

Program 6.10 Write a program to read a sorted list of floating point values and then calculate and display the median of the values.

```
using namespace std;
#include<iostream>
main()
{   int i, j, n;
    float median, values[10];
    cout<<"\n Enter the size of the array : ";
    cin>>n;
    cout<<"\n Enter the values : ";
    for(i = 0;i<n;i++)
        cin>>values[i];
    if(n%2 == 0)
        median = (values[n/2] + values[n/2+1])/2.0;
    else
        median = values[n/2 + 1];
    cout<<"\n MEDIAN = "<<median;
}
```

OUTPUT

```
Enter the size of the array : 5
Enter the values :
12 34 56 78 89
MEDIAN = 55.00
```

6.5.2 Insertion

Inserting an element in the array means adding a new data element in an already existing array. If the element has to be inserted at the end of the existing array, then the task of inserting is quite simple. We just have to add 1 to the upper_bound and assign the value. Here, we assume that the memory space allocated for the array is still available. For example, if an array is declared to contain 10 elements, but is currently having only 8 elements then, obviously, there is space to accommodate two more elements. However, if it already has 10 elements then we will not be able to add another element to it.

However, if we have to insert an element in the middle of the array, for example, in an array whose elements are arranged in ascending order then we have to first find the location where the new element will be inserted and then move all the elements that have a value greater than that of the new element one space to the right, so that space can be created to store the new value.

Program 6.11 Write a program to insert a number at a given location in an array.

```
using namespace std;
#include<iostream>
int main()
{   int i, n, num, pos, arr[10];
    cout<<"\n Enter the number of elements in the array : ";
    cin>>n;
    cout<<"\n Enter the elements :\n";
    for(i = 0;i < n;i++)
        cin>>arr[i];
    cout<<"\n Enter the number to be inserted : ";
    cin>>num;
    cout<<"\n Enter the position at which the number has to be added : ";
    cin>>pos;
    for(i = n;i >= pos;i--)
        arr[i + 1] = arr[i];
        arr[pos] = num;
    cout<<"\n The array after insertion of "<<num<<" is : ";
    for(i = 0;i < n+1;i++)
        cout<<"\n Arr[:"<<i<<"] = "<<arr[i];
}
```

OUTPUT

```
Enter the number of elements in the array : 5
Enter the values :
1 2 3 4 5
Enter the number to be inserted : 7
Enter the position at which the number has to be added :3
The array after insertion of 7 is :
1 2 3 7 4 5
```

Program 6.12 Write a program to insert a number in an array that is already sorted in ascending order.

```
using namespace std;
#include<iostream>
```

```
int main()
{   int i, n, j, num, arr[10];
    cout<<"\n Enter the number of elements in the array : ";
    cin>>n;
    cout<<"\n Enter the elements :\n";
    for(i = 0;i < n;i++)
        cin>>arr[i];
    cout<<"\n Enter the number to be inserted : ";
    cin>>num;
    for(i = 0;i < n;i++)
    {   if(arr[i] > num)
        {   for(j = n-1; j >= i; j--)
            arr[j+1] = arr[j];
        arr[i] = num;
        break;
        }
    }
    cout<<"\n The array after insertion of "<<num<<" is : ";
    for(i = 0;i < n+1;i++)
    cout<<"\n Arr["<<i<<"] = "<<arr[i];
}
```

OUTPUT

```
Enter the number of elements in the array : 5
1 2 3 4 5
Enter the number to be inserted : 6
The array after insertion of 6 is :
1 2 3 4 5 6
```

6.5.3 Deletion

Deleting an element from the array means removing a data element from an already existing array. If the element has to be deleted from the end of the existing array, then the task of deletion is quite simple. We just have to subtract 1 from the `upper_bound`.

However, if we have to delete the element from the middle of the array, then this task is not trivial. As on an average, we might have to move as much as half of the elements from its position in order to occupy the space of the deleted element.

For example, consider an array whose elements are arranged in ascending order. If an element has to be deleted probably from somewhere middle in the array, find the location from where the element has to be deleted and then move all the elements that have a value greater than that of the element one space towards the left so that space vacated by the deleted element be occupied by rest of the elements.

6.5.4 Merging

Merging two arrays in a third array means first copying the contents of the first array into the third array and then copying the contents of the second array into the third array. Hence, the merged array contains contents of the first array followed by the contents of the second array.

If the arrays are unsorted, then merging the arrays is very simple as one just needs to copy the contents of one array into another. However, merging is not a trivial task when two arrays are sorted and the merged array also needs to be sorted. Let us first discuss the merge operation on unsorted arrays. This operation is shown in Fig. 6.12.

Program 6.13 Write a program to merge two unsorted arrays.

```
using namespace std;
#include<iostream>
main()
{   int arr1[10], arr2[10], arr3[20];
    int i, n1, n2, m, index=0;
    cout<<"\n Enter the number of elements in array1 : ";
    cin>>n1;
    cout<<"\n\n Enter the Elements of the first array";
    cout<<"\n ***************************************\n";
    for(i = 0;i < n1;i++)
        cin>>arr1[i];
    cout<<"\n Enter the number of elements in array2 : ";
    cin>>n2;
    cout<<"\n\n Enter the Elements of the second array";
    cout<<"\n ****************************************\n";
    for(i = 0;i < n2;i++)
        cin>>arr2[i];
    m = n1 + n2;
    for(i = 0;i < n1;i++)
    {   arr3[index] = arr1[i];
        index++;
    }
    for(i = 0;i < n2;i++)
    {   arr3[index] = arr2[i];
        index++;
    }
    cout<<"\n\n The merged array is";
    cout<<"\n **********************";
    for(i = 0;i < m;i++)
        cout<<"\n Arr["<<i<<"] = "<<arr3[i];
}
```

OUTPUT

```
Enter the number of elements in array1 : 3
Enter the Elements of the first array :
***************************************
1   2   3
Enter the number of elements in array2 : 4
Enter the Elements of the second array
****************************************
4   5   6   7
```

location in the merged array.

Program 6.14 Write a program to merge two sorted arrays.

```
using namespace std;
#include<iostream>
main()
{   int arr1[10], arr2[10], arr3[20];
    int i, n1, n2, m, index=0;
    int index_first = 0, index_second = 0;
    cout<<"\n Enter the number of elements in array1 : ";
    cin>>n1;
    cout<<"\n\n Enter the Elements of the first array";
    for(i = 0;i < n1;i++)
        cin>>arr1[i];
    cout<<"\n Enter the number of elements in array2 : ";
    cin>>n2;
    cout<<"\n\n Enter the Elements of the second array";
    for(i = 0;i < n2;i++)
        cin>>arr2[i];
    m = n1 + n2;
    while(index_first < n1 && index_second < n2)
    {       if(arr1[index_first]<arr2[index_second])
        {   arr3[index] = arr1[index_first];
            index_first++;
        }
        else
        {   arr3[index] = arr2[index_second];
```

```
            index_second++;
        }
        index++;
    }
   // if elements of the first array is over and the second array has some elements
        if(index_first == n1)
        {   while(index_second<n2)
            {   arr3[index] = arr2[index_second];
                index_second++;
                index++;
            }
        }
// if elements of the second array is over and the first array has some elements
    else if(index_second == n2)
    {   while(index_first<n1)
        {   arr3[index] = arr1[index_first];
            index_first++;
            index++;
        }
    }
    cout<<"\n\n The contents of the merged array are-";
    cout<<"\n ***************************************";
    for(i = 0;i < m;i++)
        cout<<"\n Arr["<<i<<"] = "<<arr3[i];
}
```

OUTPUT

```
Enter the number of elements in array1 : 3
Enter the Elements of the first array : 1   2   3
Enter the number of elements in array2 : 4
Enter the Elements of the second array : 4   5   6   7
The merged array is: 1 2 3 4 5 6 7
```

6.5.5 Searching the Array Elements

Searching refers to find whether a particular value is present in the array or not. If the value is present in the array, then searching is said to be successful and the searching process gives the location of that value in the array. Otherwise, if the value is not present in the array, the searching process displays the appropriate message and in this case searching is said to be unsuccessful.

There are two popular methods for searching the elements of an array such as linear search and binary search. The algorithm that should be used depends entirely on how the values are organized in the array. For example, if the elements of the array are arranged in the ascending order, then binary search should be used as it is more efficient for sorted list in terms of complexity. We will discuss these two methods in detail in this section.

Linear Search

Linear search, also called sequential search, is a very simple method used for searching an array for a particular value. It works by comparing every element of the array one by one in sequence until a match is found. Linear search is mostly used to search an unordered list of elements. This refers to an array in which data elements are not sorted. For example, if an array `A[]` is declared and initialized as

```
int A[] = { 10, 8, 2, 7, 3, 4, 9, 1, 6, 5};
```

Program 6.15 Write a program to implement linear search in any given array.

```
using namespace std;
#include<iostream>
main()
{   int arr[10], num, i, n, found = 0, pos = -1;
    cout<<"\n Enter the number of elements in the array : ";
    cin>>n;
    cout<<"\n Enter the elements :";
    for(i = 0;i < n;i++)
```

Program 6.16 Write a program to implement binary search.

```
using namespace std;
#include<iostream>
main()
{   int arr[10], num, i, n, beg, end, mid, found = 0;
    cout<<"Enter the number of elements in the array";
    cin>>n;
    cout<<"\n Enter the elements :";
    for(i = 0;i < n;i++)
        cin>>arr[i];
    cout<<"\n Enter the number that has to be searched :";
    cin>>num;
    beg = 0, end = n - 1;
    while(beg <= end)
    {   mid = (beg + end)/2;
        if (arr[mid] == num)
        {   cout<<"\n"<<num<<" is present in the array at position = "<<mid;
            found = 1;
            break;
        }
        if (arr[mid]>num)
            end = mid-1;
        else if (arr[mid] < num)
            beg = mid + 1;
        }
        if ( beg > end && found == 0)
            cout<<"\n"<<num<<" DOES NOT EXIST IN THE ARRAY";
}
```

OUTPUT

```
Enter the number of elements in the array : 5
Enter the elements : 1 2 3 4 5
Enter the number that has to be searched 7
7   DOES NOT EXIST in the array
```

ments. To declare the array

Programming Tip: When an entire array is to be sent to the called function, the calling function just need to pass the name of the array.

Calling function

```
main()
{
    int arr[5] = {1, 2, 3, 4,
5};
    func(arr);
}
```

Called function

```
void func(int arr[5])
{
    int i;
    for(i = 0;i < 5;i++)
        cout<<num;
}
```

Figure 6.20 Passing entire array to function

Program 6.17 Write a program to read an array of n numbers and then find out the smallest number.

```
using namespace std;
#include<iostream>
void read_array( int arr[], int);
int find_small(int arr[], int n);
int main()
{   int num[10], n, smallest;
    cout<<"\n Enter the size of the array : ";
    cin>>n;
    read_array(num, n);
    smallest = find_small(num, n);
    cout<<"\n The smallest number in the array is = "<<smallest;
}
void read_array( int arr[10], int n)
{   int i;
    for(i = 0;i < n;i++)
        cin>>arr[i];
}
int find_small(int arr[10], int n)
{   int i, small = arr[0];
    for(i = 0;i < n;i++)
    {   if(arr[i] < small)
            small = arr[i];
    }
    return small;
}
```

OUTPUT

```
Enter the size of the array : 5
1 2 3 4 5
The smallest number in the array is = 1
```

Program 6.18 Write a program to merge two integer arrays. Display the merged array in reverse order.

```
using namespace std;
#include<iostream>
void read_array(int arr1[], int);
void display_array(int arr3[], int);
void merge_array(int arr3[], int, int arr1[], int arr2[], int );
void reverse_array(int my_array[], int);
int main()
{   int arr1[10], arr2[10], arr3[20], n, m, t;
    cout<<"\n Enter the size of the first array : ";
    cin>>m;
    read_array(arr1, m);
    cout<<"\n Enter the size of the second array : ";
    cin>>n;
    read_array(arr2, n);
    t = m + n;
    merge_array(arr3, m, arr1, arr2, n);
    cout<<"\n The merged array is : ";
    display_array(arr3, t);
    cout<<"\n The merged array in reverse order is : ";
    reverse_array(arr3, t);
}
void read_array( int my_array[10], int n)
{   int i;
    for(i = 0;i < n;i++)
    cin>>my_array[i];
}
void merge_array(int my_array1[], int m, int my_array2[], int my_array3[], int n)
{   int i, j = 0;
    for(i = 0; i < m; i++)
    {   my_array1[j] = my_array2[i];
        j++;
    }
    for(i = 0; i < n; i++)
    {   my_array1[j] = my_array3[i];
        j++;
    }
}
void display_array( int my_array[], int n)
{   int i;
    for(i = 0;i < n;i++)
        cout<<"\n array["<<i<<"d] = "<<my_array[i];
}
void reverse_array(int my_array[], int m)
{   int i, j;
    for(i = m-1, j = 0;i >= 0;i--, j++)
        cout<<"\n array["<<i<<"] = "<<my_array[i];
}
```

OUTPUT

```
Enter the number of elements in array1 : 3
Enter the Elements of the first array : 1   2   3
Enter the number of elements in array2 : 4
Enter the Elements of the second array : 4   5   6   7
The merged array is: 1 2 3 4 5 6 7
The merged array in reverse order is: 7 6 5 4 3 2 1
```

Program 6.19 Write a program to interchange the biggest and the smallest number in the array using functions.

```
using namespace std;
#include<iostream>
void read_array(int arr[], int);
void display_array(int arr[], int);
void interchange(int arr[], int);
int find_biggest_pos(int my_array[10], int n);
int find_smallest_pos(int my_array[10], int n);
int main()
{   int arr[10], n;
    cout<<"\n Enter the size of the array : ";
    cin>>n;
    read_array(arr, n);
    interchange(arr, n);
}
void read_array( int my_array[10], int n)
{   int i;
    for(i = 0;i < n;i++)
        cin>>my_array[i];
}
void display_array( int my_array[10], int n)
{   int i;
    for(i = 0;i < n;i++)
        cout<<"\n array["<<i<<"] = "<<my_array[i];
}
void interchange(int my_array[10], int n)
{   int temp, big_pos, small_pos;
    big_pos = find_biggest_pos(my_array, n);
    small_pos = find_smallest_pos(my_array,n);
    temp = my_array[big_pos];
    my_array[big_pos] = my_array[small_pos];
    my_array[small_pos] = temp;
    display_array(my_array,n);
}
int find_biggest_pos( int my_array[10], int n)
{   int i, large, pos = -1;
    large = my_array[0];
    for(i = 1;i < n;i++)
    {   if (my_array[i] > large)
        {   large = my_array[i];
            pos = i;
        }
    }
    return pos;
}
int find_smallest_pos( int my_array[10], int n)
{   int i, small , pos = 0;
    small = my_array[0];
```

```
    for(i = 1;i < n;i++)
    {   if (my_array[i] < small)
        {   small = my_array[i];
            pos = i;

        }

    }
    return pos;

}
```

OUTPUT

```
Enter the number of elements in array1 : 5
Enter the Elements of the first array : 1  2 3 4 5
5 2 3 4 1
```

Note If a function receives an array that does not change it, then the array should be received as a constant array. This would ensure that the contents of the array are not accidentally changed. To declare an array as constant, prefix its type with the const keyword, as.

```
int sum(const int arr[], int n);
```

6.7 TWO-DIMENSIONAL ARRAYS

Till now, we have read only about one-dimensional arrays. A one-dimensional array is organized linearly in only in one direction. At times, we need to store data in the form of matrices or tables. Here, the concept of single dimension arrays is extended to incorporate two-dimensional data structures. A two-dimensional array is specified using two subscripts where one subscript denotes row and the other denotes column. A two-dimensional array is an array of a one-dimensional array. Figure 6.21 shows a 2D array which can be viewed as array of arrays.

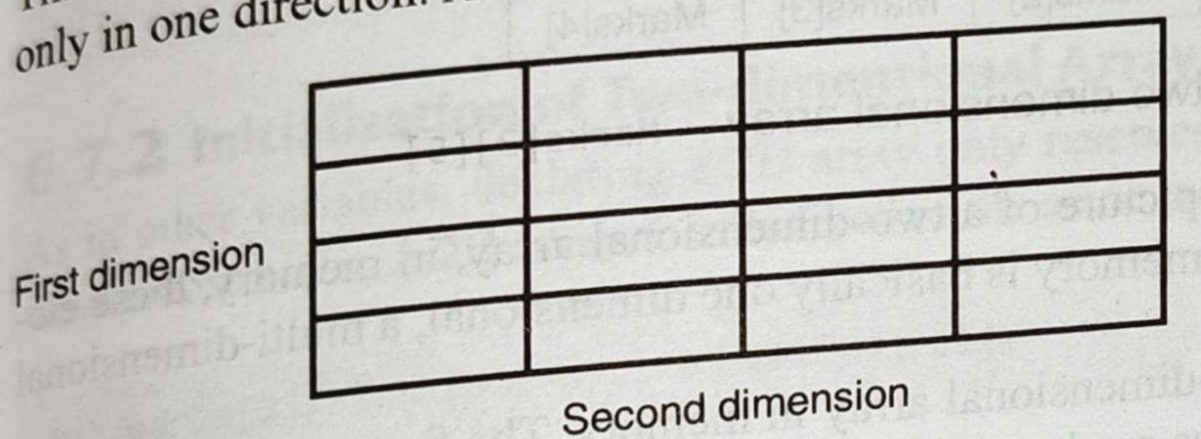


Figure 6.21 Two-dimensional array

6.7.1 Declaration of Two-dimensional Arrays

Like one-dimensional array, two-dimensional arrays must be declared before being used. The declaration statement tells the compiler the name of the array, the data type of each element in the array, and the size of each dimension. A 2D array is declared as follows:

```
data_type array_name[row_size][column_size];
```

Therefore, a two-dimensional m × n array contains m*n data elements and each element is accessed using two subscripts, i and j, where i<=m and j<=n.

For example, if we want to store the marks obtained by three students in five different subjects, then we can declare a two-dimensional array as follows:

```
int marks[3][5]
```

Program 6.20 Write a program to print the elements of a 2D array.

```
using namespace std;
#include<iostream>
main()
{   int arr[2][2] = {12, 34, 56,32};
    int i, j;
    for(i = 0;i < 2;i++)
    {   cout<<"\n";
        for(j = 0;j < 2;j++)
            cout<<"\t"<<arr[i][j];
    }
}
```

OUTPUT

```
12    34
56    32
```

Program 6.21 Write a program to generate Pascal's Triangle.

```
using namespace std;
#include<iostream>
main()
{   int arr[7][7]={0};
    int row = 2, col, i, j;
    arr[0][0] = arr[1][0] = arr[1][1] = 1;
    while(row <= 7)
    {   arr[row][0] = 1;
        for(col = 1;col <= row;col++)
            arr[row][col] = arr[row-1][col-1] + arr[row-1][col];
            row++;
    }
    for(i = 0;i < 7;i++)
    {   cout<<"\n";
        for(j = 0;j <= i;j++)
            cout<<"\t"<<arr[i][j];

    }
}
```

OUTPUT

```
1
1   1
1   2   1
1   3   3    1
1   4   6    4    1
1   5   10   10   5   1
1   6   15   20   15  6    1
```

Program 6.22 In a small company, there are five 5 salesmen. Each salesman is supposed to sell 3 products. Write a program using two- dimensional array to print (i) the total sales by each salesman and (ii) total sales of each item.

```
using namespace std;
#include<iostream>
main()
{   int sales[5][3], i, j, total_sales=0;
    //INPUT DATA
cout<<"\n ENTER THE DATA";
for(i = 0;i < 5;i++)
{   cout<<"\n Enter the sales of 3 items sold by "<<i<<"salesman : ";
    for(j = 0;j < 3;j++)
        cin>>sales[i][j];
}
// PRINT TOTAL SALES BY EACH SALESMAN
for(i = 0;i < 5;i++)
{   total_sales = 0;
    for(j = 0;j < 3;j++)
        total_sales += sales[i][j];
    cout<<"\n Total Sales By "<<i<<" Salesman = "<<total_sales;
```

```
    }
    // TOTAL SALES OF EACH ITEM
    for(i = 0;i < 3;i++)                    // for each item
    {   total_sales=0;
        for(j = 0;j < 5;j++)                // for each salesman
            total_sales += sales[j][i];
        cout<<"\n Total sales of "<<i<<" item = "<<total_sales;
    }
}
```

OUTPUT

```
Enter the sales of 3 items sold by 0 salesman = 56 67 78
Enter the sales of 3 items sold by 1 salesman = 78 89 90
Enter the sales of 3 items sold by 2 salesman = 12 23 34
Enter the sales of 3 items sold by 3 salesman = 45 56 67
Total Sales By 0 Salesman = 135
Total Sales By 1 Salesman = 201
Total Sales By 2 Salesman = 257
Total Sales By 3 Salesman = 69
Total Sales By 3 Salesman = 168
Total sales of 0 item = 225
Total sales of 1 item = 280
Total sales of 2 item = 325
```

Program 6.23 In a class, there are 10 students. Each student is supposed to appear in three tests. Write a program using two dimensional array to print the marks obtained by each student in different subjects, total marks and average obtained by each student, and store the average of each student in a separate 1D array so that it can be used to calculate the class average.

```
using namespace std;
#include<iostream>
main()
{   int marks[10][3], i, j;
    int total_marks[10] = {0};;
    float class_avg = 0.0, total_avg = 0.0;
    float avg[10];
    //INPUT DATA
    cout<<"\n ENTER THE DATA";
    for(i = 0;i < 10;i++)
    {   cout<<"\n Enter the marks of "<<i<<" student in 3 subjects : ";
        for(j = 0;j < 3;j++)
            cin>>marks[i][j];
}
// CALCULATE TOTAL MARKS OF EACH STUDENT
for(i = 0;i < 10;i++)
{   for(j = 0;j < 3;j++)
        total_marks[i] += marks[i][j];
}
// CALCULATE AVERAGE OF EACH STUDENT
for(i = 0;i < 10;i++)
{   for(j = 0;j < 3;j++)
    avg[i] = (float)total_marks[i]/3.0;
```

```
}
// CALCULATE CLASS AVERAGE
for(i = 0;i < 10;i++)
    total_avg += avg[i];
class_avg = (float)total_avg/10;
// DISPLAY RESULTS
cout<<"\n\n STUD NO. \t MARKS OBTAINED IN THREE SUBJECTS \t TOTAL MARKS \t AVERAGE";
cout<<"\n\t*************************************************************";
for(i = 0;i < 10;i++)
{   cout.width(4);
    cout.precision(2);
    cout<<"\n"<<i;
    for(j = 0;j < 3;j++)
        cout<<"\t\t"<<marks[i][j];
        cout<<"\t\t"<<total_marks[i]<<"\t"<<avg[i];
    }
    cout<<"\n\n CLASS AVERAGE = "<<class_avg;
}
```

OUTPUT

```
ENTER THE DATA
Enter the marks of student 0 in 3 subjects: 78 89 90
Enter the marks of student 1 in 3 subjects: 98 87 76
Enter the marks of student 2 in 3 subjects: 67 78 89
Enter the marks of student 3 in 3 subjects: 90 87 65
Enter the marks of student 4 in 3 subjects: 56 87 97
Enter the marks of student 5 in 3 subjects: 45 67 89
Enter the marks of student 6 in 3 subjects: 66 77 88
Enter the marks of student 7 in 3 subjects: 76 87 98
Enter the marks of student 8 in 3 subjects: 67 88 66
Enter the marks of student 9 in 3 subjects: 66 75 78

 STUD NO       MARKS IN 3 SUBJECTS      TOTAL MARKS  AVERAGE
 0             78    89    90           257          85.67
 1             98    87    76           261          87.00
 2             67    78    89           234          78.00
 3             90    87    65           242          80.67
 4             56    87    97           240          80.00
 5             45    67    89           201          67.00
 6             66    77    88           231          77.00
 7             76    87    98           261          87.00
 8             67    88    66           221          73.67
 9             66    75    78           210          73.00
 CLASS AVERAGE = 79
```

Program 6.24 Write a program to read the marks in a two-dimensional array which stores marks of 10 students in five subjects. Write a program to display the highest marks in each subject.

```
using namespace std;
#include<iostream>
main()
{   int marks[10][5], i, j, max_marks;
    for(i = 0;i < 10;i++)
    {   cout<<"\n Enter the marks obtained by "<<i<<"th student";
```

```
    for(j = 0;j < 5;j++)
    {   cout<<"\n marks["<<i<<"]["<<j<<"] = ";
        cin>>marks[i][j];
    }
}
for(j = 0;j < 5;j++)
{   max_marks = -999;
    for(i = 0;i < 10;i++)
    {   if(marks[i][j]>max_marks)
        max_marks = marks[i][j];
    }
cout<<"\n The highest marks obtained in the "<<j<<" subject = "<<marks[i][j];
}
```

SAMPLE OUTPUT

```
The highest marks obtained in the 0 subject = 98
The highest marks obtained in the 1 subject = 94
The highest marks obtained in the 2 subject = 97
The highest marks obtained in the 3 subject = 89
```

6.8 OPERATIONS ON TWO-DIMENSIONAL ARRAYS

We can perform the following operations on a two-dimensional array:

Transpose Transpose of a $m \times n$ matrix A is given as a $n \times m$ matri $\times$ B where,

$$B_{i,j} = A_{j,i}$$

Sum Two matrices that are compatible with each other can be added together, thereby, storing the result in the third matrix. Two matrices are said to be compatible when they have the same number of rows and columns. The elements of the matrices can be added by writing as follows:

$$C_{i,j} = A_{i,j} + B_{i,j}$$

Difference Two matrices that are compatible with each other can be subtracted, thereby, storing the result in the third matrix. The elements of the matrices can be subtracted by writing as follows:

$$C_{i,j} = A_{i,j} - B_{i,j}$$

Product Two matrices can be multiplied with each other if the number of columns in the first matrix is equal to the number of rows of the second matrix. Therefore, $m \times n$ matrix A can be multiplied with a $p \times q$ matrix if $n = q$. The elements of the matrices can be multiplied by writing as follows:

$$C_{i,j} = \sum A_{i},kB_{j},k, \text{ for } k = 1 \text{ to } k<n$$

Program 6.25 Write a program to transpose a 3 × 3 matrix.

```
using namespace std;
#include<iostream>
```

```
int main()
{   int i, j, mat[3][3], transposed_mat[3][3];
    cout<<"\n Enter the elements of the matrix ";
    for(i = 0;i < 3;i++)
    {   for(j = 0;j < 3;j++)
        {   cout<<"\n mat["<<i<<"]["<<j<<"] = ";
            cin>>mat[i][j];
        }
    }
    cout<<"\n The elements of the matrix are ";
    for(i = 0;i < 3;i++)
    {   cout<<"\n";
        for(j = 0;j < 3;j++)
            cout<<"\t mat["<<i<<"]["<<j<<"] = "<<mat[i][j];
    }
    for(i = 0;i < 3;i++)
    {   for(j = 0;j < 3;j++)
            transposed_mat[i][j] = mat[j][i];
    }
    cout<<"\n The elements of the transposed matrix are ";
    for(i = 0;i < 3;i++)
    {   cout<<"\n";
        for(j = 0;j < 3;j++)
        cout<<"\t transposed_mat["<<i<<"]["<<j<<"] = "<< transposed_mat[i][j];
    }
}
```

OUTPUT

```
Enter the elements of the matrix
1 2 3 4 5 6 7 8 9
The elements of the matrix are
1 2 3
4 5 6
7 8 9
The elements of the transposed matrix are
1 4 7
2 5 8
7 8 9
```

Program 6.26 Write a program to multiply two m × n matrices.

```
using namespace std;
#include<iostream>
#include<stdlib.h>
int main()
{   int i, j, k;
    int rows1, cols1, rows2, cols2, res_rows, res_cols;
    int mat1[5][5], mat2[5][5], res[5][5];
    cout<<"\n Enter the numbers of rows and columns in the first matrix : ";
    cin>>rows1>>cols1;
    cout<<"\n Enter the numbers of rows and columns in the second
    matrix : ";
    cin>>rows2>>cols2;
    if(cols1 != rows2)
```

```
int main()
{   int i, j, mat[3][3], transposed_mat[3][3];
    cout<<"\n Enter the elements of the matrix ";
    for(i = 0;i < 3;i++)
    {   for(j = 0;j < 3;j++)
        {   cout<<"\n mat["<<i<<"]["<<j<<"] = ";
            cin>>mat[i][j];
        }
    }
    cout<<"\n The elements of the matrix are ";
    for(i = 0;i < 3;i++)
    {   cout<<"\n";
        for(j = 0;j < 3;j++)
            cout<<"\t mat["<<i<<"]["<<j<<"] = "<<mat[i][j];
    }
    for(i = 0;i < 3;i++)
    {   for(j = 0;j < 3;j++)
            transposed_mat[i][j] = mat[j][i];
    }
    cout<<"\n The elements of the transposed matrix are ";
    for(i = 0;i < 3;i++)
    {   cout<<"\n";
        for(j = 0;j < 3;j++)
        cout<<"\t transposed_mat["<<i<<"]["<<j<<"] = "<< transposed_mat[i][j];
    }
}
```

OUTPUT

```
Enter the elements of the matrix
1 2 3 4 5 6 7 8 9
The elements of the matrix are
1 2 3
4 5 6
7 8 9
The elements of the transposed matrix are
1 4 7
2 5 8
7 8 9
```

Program 6.26 Write a program to multiply two m × n matrices.

```
using namespace std;
#include<iostream>
#include<stdlib.h>
int main()
{   int i, j, k;
    int rows1, cols1, rows2, cols2, res_rows, res_cols;
    int mat1[5][5], mat2[5][5], res[5][5];
    cout<<"\n Enter the numbers of rows and columns in the first matrix : ";
    cin>>rows1>>cols1;
    cout<<"\n Enter the numbers of rows and columns in the second
    matrix : ";
    cin>>rows2>>cols2;
    if(cols1 != rows2)
```

```
{   cout<<"\n The number of columns in the first matrix must be equal
    to the number of rows in the second matrix";
    exit(1);
}
res_rows = rows1, res_cols = cols2;
cout<<"\n Enter the elements of the first matrix ";
for(i = 0;i < rows1;i++)
{   for(j = 0;j < cols1;j++)
    {   cout<<"\n mat1["<<i<<"]["<<j<<"] = ";
        cin>>mat1[i][j];
    }
}
cout<<"\n Enter the elements of the second matrix ";
for(i = 0;i < rows2;i++)
{   for(j = 0;j < cols2;j++)
{   cout<<"\n mat1["<<i<<"]["<<j<<"] = ";
    cin>>mat2[i][j];
    }
}
for(i = 0;i < res_rows;i++)
{   j = 0;
    for(j = 0;j < res_cols;j++)
    {   res[i][j] = 0;
        for(k = 0; k < cols1;k++)
            res[i][j] += mat1[i][k] * mat2[k][j];
        }
    }
    cout<<"\n The elements of the product matrix are ";
    for(i = 0;i < res_rows;i++)
    {   cout<<"\n";
        for(j = 0;j < res_cols;j++)
            cout<<"\t res["<<i<<"]["<<j<<"] = "<<res[i][j];
    }
}
```

OUTPUT

```
Enter the numbers of rows and columns in the first matrix : 2 2
Enter the numbers of rows and columns in the second matrix : 2 2
Enter the elements of the first matrix 1 2 3 4
Enter the elements of the second matrix 5 6 7 8
The elements of the product matrix are 19 22 43 50
```

6.9 TWO-DIMENSIONAL ARRAYS FOR INTER-FUNCTION COMMUNICATION

There are three ways of passing parts of the two-dimensional array to a function. First, we can pass individual elements of the array. This is done in exactly the same way as we passed the element of a one-dimensional array. Second, we can pass a single row of the two-dimensional array. This is equivalent to passing the entire one-dimensional array to a function that has already been discussed in the previous section. Third, we can

pass the entire two- dimensional array to the function. Refer to Fig. 6.26, which shows three ways of using two-dimensional array for inter-function communication.

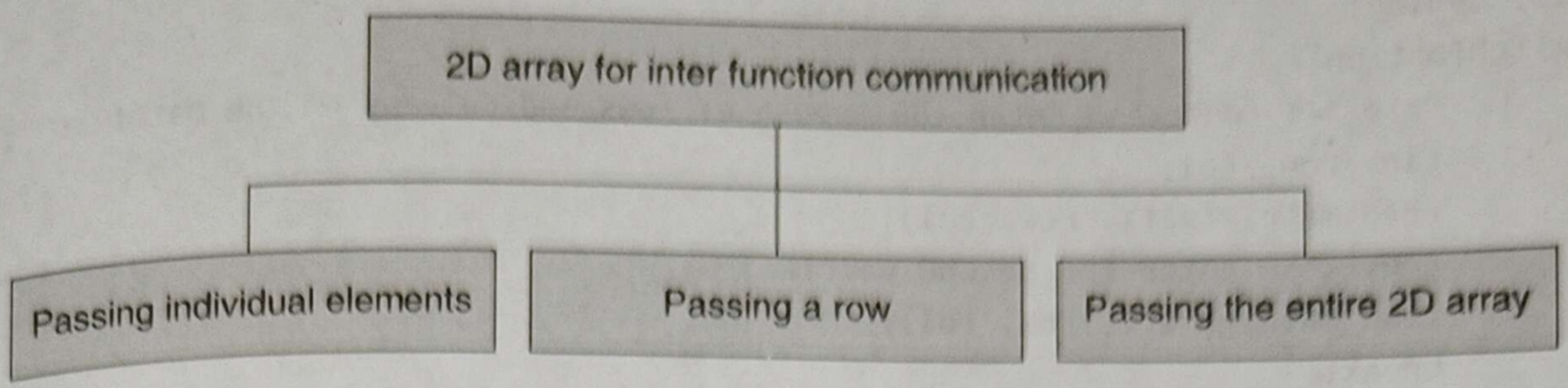


Figure 6.26 Two-dimensional arrays for inter-function communication

```
Calling Function
main()
{       int arr[2][3]= ({1, 2, 3}, {4, 5, 6} };
        func(arr[1]);

}
                Called Function
void func(int arr[])
{       int i;
        for(i=0;i<5;i++)
            cout<<arr[i] * 10;

}
```

Figure 6.27 Passing two-dimensional arrays for inter function communication

6.9.1 Passing a Row

A row of a two-dimensional array can be passed by indexing the array name with the row number. When we send a single row of a two dimensional array, then the called function receives a one dimensional array. Figure 6.27 illustrates how a single row of a two-dimensional array is passed to the called function.

6.9.2 Passing the Entire Two-dimensional Array

To pass a two dimensional array to a function, we use the array name as the actual parameter, as done in the case of a 1D array. However, the parameter in the called function must indicate that the array has two dimensions.

Programming Tip: A compiler error will be generated if you omit the array size in the parameter declaration for any array dimension other than the first.

Program 6.27 Write a menu driven program to read and display an m × n matrix. Find the sum, transpose, and product of two m × n matrices.

```
using namespace std;
#include<iostream>
void read_matrix(int mat[5][5], int, int);
void sum_matrix(int mat1[5][5], int mat2[5][5], int, int);
void mul_matrix(int mat1[5][5], int mat2[5][5], int, int);
void transpose_matrix(int mat2[5][5], int, int);
void display_matrix(int mat[5][5], int r, int c);
int main()
{   int option, row, col;
    int mat1[5][5], mat2[5][5];
    do
    {   cout<<"\n ******* MAIN MENU ********";
        cout<<"\n 1. Read the two matrices";
        cout<<"\n 2. Add the matrices";
        cout<<"\n 3. Multiply the matrices";
        cout<<"\n 4. Transpose the matrix";
```

```
cout<<"\n 5. EXIT";
cout<<"\n\n Enter your option : ";
cin>>option;
switch(option)
    {   case 1 : cout<<"\n Enter the number of rows and columns of the matrix : ";
        cin>>row>>col;
        read_matrix(mat1, row, col);
        cout<<"\n Enter the second matrix : ";
        read_matrix(mat2, row, col);
        break;
        case 2 : sum_matrix(mat1, mat2, row, col);
        break;
        case 3 : if(col == row)
            mul_matrix(mat1, mat2, row, col);
        else
            cout<<"\n To multiply two matrices, number of columns in the
            first matrix must be equal to number of rows in the second matrix";
        break;
        case 4:
            transpose_matrix(mat1, row, col);
            transpose_matrix(mat2, row, col);
        break;
    }
}while(option != 5);
}
void read_matrix(int mat[5][5], int r, int c)
{   int i, j;
    for(i = 0;i < r;i++)
    {   for(j = 0;j < c;j++)
        {   cout<<"\n mat["<<i<<"]["<<j<<"] = ";
            cin>>mat[i][j];
        }
    }
}
void sum_matrix(int mat1[5][5], int mat2[5][5], int r, int c)
{   int i, j, sum[5][5];
    for(i = 0;i < r;i++)
    {   for(j = 0;j < c;j++)
        sum[i][j] = mat1[i][j] + mat2[i][j];
    }
    display_matrix(sum, r, c);
}
void mul_matrix(int mat1[5][5], int mat2[5][5], int r, int c)
{   int i, j, k, prod[5][5];
    for(i = 0;i < r;i++)
    {   for(j = 0;j < c;j++)
        {   prod[i][j] = 0;
            for(k = 0;k < col;k++)
                prod[i][j] += mat1[i][k] * mat2[k][j];
```

```
    }
  }
  display_matrix(prod, r, c);

}
void transpose_matrix(int mat[5][5], int r, int c)
{  int i, j, tp_mat[5][5];
   for(i = 0;i < r;i++)
   {  for(j = 0;j < c;j++)
      {
         tp_mat[i][j] = mat[j][i];

      }
   }
   display_matrix(mat, r, c);
}
void display_matrix(int mat[5][5], int r, int c)
{  int i, j;
   for(i = 0;i < r;i++)
   {  cout<<"\n";
      for(j = 0;j < c;j++)
         cout<<"\t mat["<<i<<"]["<<"j="<<mat[i][j];

   }
}
```

OUTPUT

```
******* MAIN MENU ********
1. Read the two matrices
2. Add the matrices
3. Multiply the matrices
4. Transpose the matrix
5. EXIT
Enter your option: 1
Enter the number of rows and columns of the matrix: 2 2
Enter the first matrix:
mat[0][0] = 1 mat[0][1] = 2
mat[1][0] = 3 mat[1][1] = 4
Enter the second matrix :
mat[0][0] = 2 mat[0][1] = 3
mat[1][0] = 4 mat[1][1] = 5
```

Program 6.28 Write a program to fill a square matrix with value zero on the diagonals, 1 on the upper right triangle and -1 on the lower left triangle.

```
using namespace std;
#include<iostream>
void read_matrix(int mat[5][5], int, int);
void sum_matrix(int mat1[5][5], int mat2[5][5], int, int);
void mul_matrix(int mat1[5][5], int mat2[5][5], int, int);
void transpose_matrix(int mat2[5][5], int, int);
void display_matrix(int mat[5][5], int r, int c);
int main()
{  int option, row, col;
   int mat1[5][5], mat2[5][5];
   do
   {  cout<<"\n ******* MAIN MENU ********";
      cout<<"\n 1. Read the two matrices";
```

```
    cout<<"\n 2. Add the matrices";
    cout<<"\n 3. Multiply the matrices";
    cout<<"\n 4. Transpose the matrix";
    cout<<"\n 5. EXIT";
    cout<<"\n\n Enter your option : ";
    cin>>option;
    switch(option)
{   case 1:cout<<"\n Enter the number of rows and columns of the matrix : ";
           cin>>row>>col;
           read_matrix(mat1, row, col);
           cout<<"\n Enter the second matrix : ";
           read_matrix(mat2, row, col);
           break;
case 2 : sum_matrix(mat1, mat2, row, col);
           break;
case3 : if(col == row)
           mul_matrix(mat1, mat2, row, col);
           else
           cout<<"\n To multiply two matrices, number of columns in the
           first matrix must be equal to number of rows in the second matrix";
           break;
case 4:
           transpose_matrix(mat1, row, col);
           transpose_matrix(mat2, row, col);
           break;
}
    }while(option != 5);
}
void read_matrix(int mat[5][5], int r, int c)
{   int i, j;
    for(i = 0;i < r;i++)
    {   for(j = 0;j < c;j++)
        {   cout<<"\n mat["<<i<<"]["<<j<<"] = ";
            cin>>mat[i][j];
        }
    }
}
void sum_matrix(int mat1[5][5], int mat2[5][5], int r, int c)
{   int i, j, sum[5][5];
    for(i = 0;i < r;i++)
    {   for(j = 0;j < c;j++)
            sum[i][j] = mat1[i][j] + mat2[i][j];
    }
    display_matrix(sum, r, c);
}
void mul_matrix(int mat1[5][5], int mat2[5][5], int r, int c)
{   int i, j, k, prod[5][5];
    for(i = 0;i < r;i++)
    {   for(j = 0;j < c;j++)
        {   prod[i][j] = 0;
            for(k = 0;k < c;k++)
                prod[i][j] += mat1[i][k] * mat2[k][j];
        }
    }
    display_matrix(prod, r, c);
```

```
}
void transpose_matrix(int mat[5][5], int r, int c)
{   int i, j, tp_mat[5][5];
    for(i = 0;i < r;i++)
    {   for(j = 0;j < c;j++)
            tp_mat[i][j] = mat[j][i];

    }
   display_matrix(mat, r, c);
}
void display_matrix(int mat[5][5], int r, int c)
{   int i, j;
    for(i = 0;i < r;i++)
    {   cout<<"\n";
        for(j = 0;j < c;j++)
            cout<<"\t "<< mat[i][j];

    }
}
```

OUTPUT

```
Enter the number of rows and columns of the matrix : 2 2
1 2 3 4
1 2
3 4
```

6.10 MULTI-DIMENSIONAL ARRAYS

A multi-dimensional array, in simple terms, is an array of arrays. Like we have one index in a single dimensional array and two indices in a two-dimensional array, we have n indices in an n-dimensional array or a multi-dimensional array. Conversely, an n dimensional array is specified using n indices. An n dimensional $m_1 \times m_2 \times m_3 \times \ldots m_n$ array is a collection $m_1 * m_2 * m_3 * \ldots * m_n$ elements. In a multi-dimensional array, a particular element is specified by using n subscripts as $A[I_1][I_2][I_3]'[I_n]$, where

$I_1 <= M_1 I_2 <= M_2 \qquad I_3 <= M_3 \qquad \ldots\ldots \qquad I_n <= M_n$

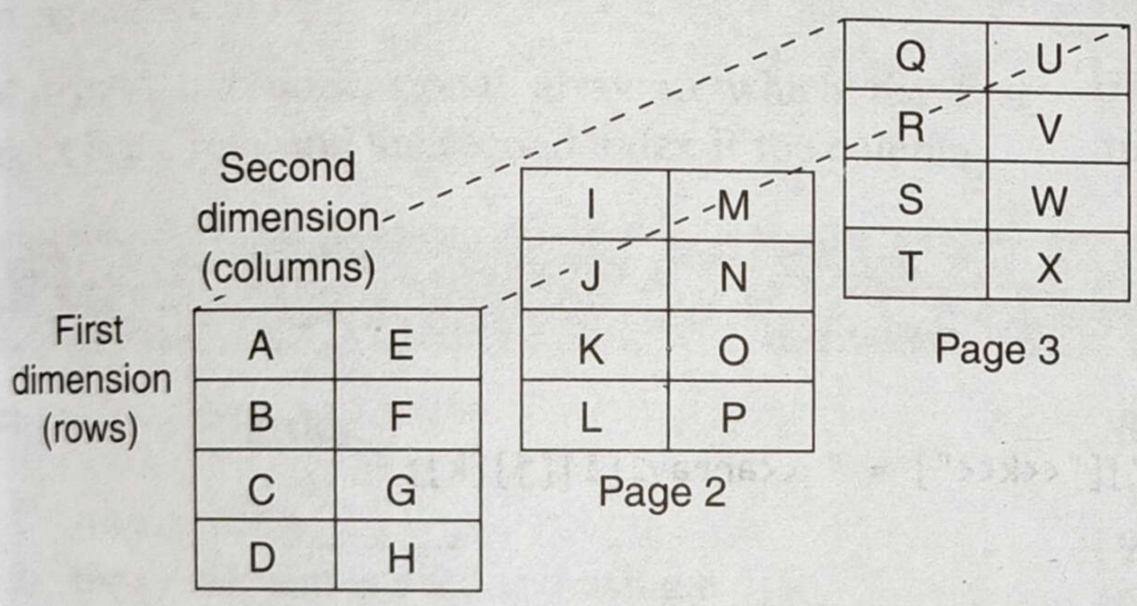


Figure 6.28 Three-dimensional array

A multi-dimensional array can contain as many indices as needed and the requirement of the memory increases with the number of indices used. However, practically speaking, we will hardly use more than three indices in any program. Figure 6.28 shows a three-dimensional array. The array has three pages, four rows, and two columns.

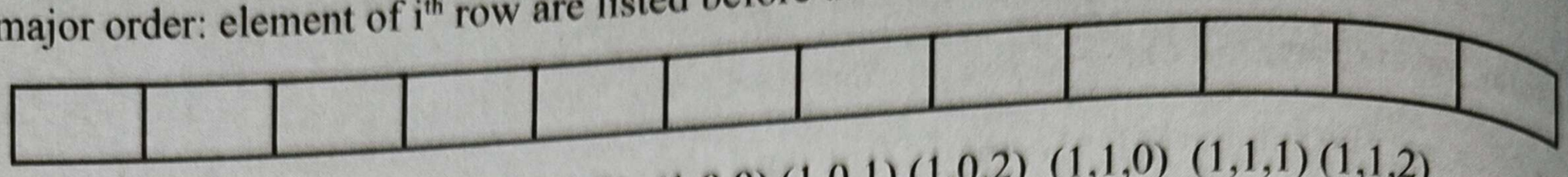
A multi-dimensional array is declared and initialized similar to one and two dimensional arrays.

Note A three dimensional array consists of pages. Each page in turn contains m rows and n columns.

Example 6.6 Consider a three-dimensional array defined as int A[2][3][2]. Calculate the number of elements in the array. Give the memory representation of the array in row major order and column major order.

(a) Row major order: Write the elements row by row.

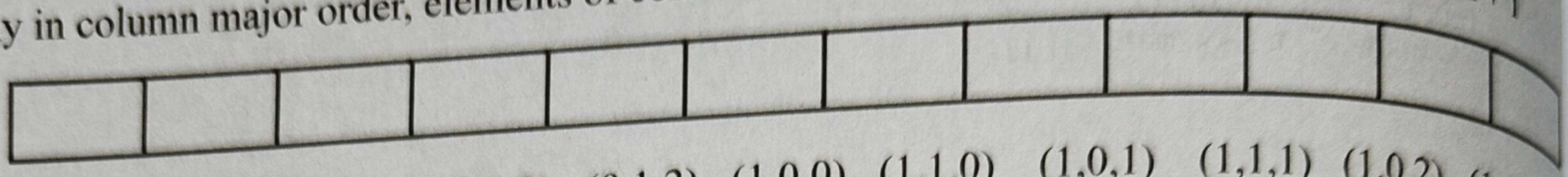
In row major order: element of i^{th} row are listed before the elements of row i + 1



(0,0,0) (0,0,1) (0,0,2) (0,1,0) (0,1,1) (0,1,2) (1,0,0) (1,0,1) (1,0,2) (1,1,0) (1,1,1) (1,1,2)

(b) Column major order: Write the elements column by column

Similarly in column major order, elements of column i are listed before the elements of column i + 1



(0,0,0) (0,1,0) (0,0,1) (0,1,1) (0,0,2) (0,1,2) (1,0,0) (1,1,0) (1,0,1) (1,1,1) (1,0,2) (1,1,2)

The three-dimensional array will contain 2*3*2 = 12 elements

Program 6.29 Write a program to read and display a 2 × 2 × 2 array.

```
using namespace std;
#include<iostream>
int main()
{   int array1[3][3][3], i, j, k;
    cout<<"\n Enter the elements of the matrix";
    for(i = 0;i < 2;i++)
    {   for(j = 0;j < 2;j++)
        {   for(k = 0;k < 2;k++)
            {   cout<<"\n array["<<i<<"]["<<j<<"]["<<k<<"] = ";
                cin>>array1[i][j][k];
            }
        }
}
cout<<"\n The matrix is : ";
for(i = 0;i < 2;i++)
{   cout<<"\n\n";
    for(j = 0;j < 2;j++)
    {   cout<<"\n";
        for(k = 0;k < 2;k++)
            cout<<"\t array["<<i<<"]["<<j<<"]["<<k<<"] = " <<array1[i][j][k];
        }
    }
}
```

OUTPUT

```
Enter the elements of the matrix
1 2 3 4 5 6 7 8
arr[0][0][0] = 1   arr[0][0][1] = 2
arr[0][1][0] = 3   arr[0][1][1] = 4
arr[1][0][0] = 5   arr[1][0][1] = 6
arr[1][1][0] = 7   arr[1][1][1] = 8
```

Points to Remember

- An array is a collection of similar data elements of the same data type.
- The elements of the array are stored in consecutive memory locations and are referenced by an index (also known as the subscript). Subscript is an ordinal number which is used to identify an element of the array.
- Declaring an array means specifying three things: data type, name, and its size.
- The name of the array is a symbolic reference to the address of the first byte of the array. Therefore, whenever we use the array name, we are actually referring to the first byte of that array. The index specifies an offset from the beginning of the array to the element being referenced.
- A two-dimensional array is specified using two subscripts where first subscript denotes row and the second denotes column. C++ considers the two-dimensional array as an array of one-dimensional arrays.
- A multidimensional array is an array of arrays. Like we have one index in a one-dimensional array, two indices in a two-dimensional array, in the same way we have *n* indices in an *n*-dimensional or multidimensional array.
- Conversely, an n-dimensional array is specified using n indices.

Glossary

Array An array is a collection of similar data elements

Array index Location of an item in an array

Binary search Searches a sorted array by repeatedly dividing the search interval in half. Begin with an interval covering the whole array. If the value of the search key is less than the item in the middle of the interval, narrow the interval to the lower half; else, narrow it to the upper half. Check until the value is found or the interval is empty.

Linear search Searches an array by checking elements one at a time

Lower triangular matrix A matrix A_{ij} forms a lower triangular when $i \geq j$

Matrix A two-dimensional array in which the first index is the row, and the second index is the column

Multi-dimensional or ***n*-dimensional:** An array with exactly *n* dimensions.

One-dimensional array An array with one dimension or one subscript or index

Rectangular matrix An n × m matrix whose size may not be the same in both dimensions

Sparse matrix A matrix that has relatively few non-zero elements

Two-dimensional array An array with two dimensions or two subscripts or indices

Three-dimensional array An array with three dimensions or three subscripts or indices

Upper triangular matrix A matrix A_{ij} forms an upper triangular when $i \leq j$

Exercises

Fill in the Blanks

1. An array is a _____.
2. Every element is accessed using a _____.
3. An expression that evaluates to a _____ value may be used as an index.
4. The elements of an array are stored in _____ memory locations.
5. An 'n' dimensional array contains _____ subscripts.
6. The name of the array acts as a _____.
7. Declaring an array means specifying the _____, _____, and _____.
8. The subscript or the index represents the offset from the beginning of the array to _____.
9. _____ is used to access an element in the array.
10. _____ is the address of the first element in the array.
11. The length of the array is given by the number of _____.
12. _____ means accessing each element of the array for a specific purpose.
13. Performance of the linear search algorithm can be improved by using a _____.
14. An array occupies _____ memory locations.

15. An array name is often known to be a _____ pointer.
16. A multi-dimensional array, in simple terms, is an _____
17. `arr[3] = 10` ; initializes the _____ element of the array with value 10.
18. The ___ search locates the value by starting at the beginning of the array and moving towards its end.

State True or False

1. An array is used to refer to multiple memory locations having the same name.
2. An array need not be declared before being used.
3. An array contains elements of the same data type.
4. A loop is used to access all the elements of the array.
5. All the elements of the array are automatically initialized to zero when the array is defined.
6. An array stores all its data elements in non-consecutive memory location.
7. To copy an array, you must copy the value of every element of the first array into the element of the second array.
8. Lower bound is the index of the last element in the array.
9. A merged array contains contents of the first array followed by the contents of the second array.
10. Binary search is also called sequential search.
11. Linear search is performed on a sorted array.
12. Binary search is the best searching algorithm for all types of arrays.
13. It is possible to pass entire array as a function argument.
14. When an array is passed to a function, the value for each element.
15. When an array is passed to a function, it is always passed by call-by-reference method.
16. Linear search can be used to search a value in any array.
17. Linear search is recommended for small arrays.
18. A two-dimensional array is nothing but an array of one-dimensional arrays.
19. A two-dimensional array contains data of two different types.
20. When an array is declared, its elements are automatically initialized to zero.
21. A char type variable can be used as a subscript in an array.
22. By default, the first subscript of the array is zero.
23. A long int value can be used as an array subscript.
24. An array can have a maximum of four dimensions.

Multiple Choice Questions

1. If an array is declared as `arr[] = {1,3,5,7,9}`, what is the value of `sizeof(arr[3])`?
   (a) 10 (b) 20
   (c) 30 (d) 40
2. If an array is declared as `arr[] = {1,3,5,7,9}`, what is the value of `arr[3]`?
   (a) 1 (b) 7
   (c) 9 (d) 5
3. If an array is declared as double `arr[50]`, how many bytes will be allocated to it?
   (a) 50 (b) 100
   (c) 200 (d) 400
4. If an array is declared as int `arr[50]`, how many elements can it hold?
   (a) 49 (b) 50
   (c) 51 (d) 0
5. If an array is declared as int `arr[5][5]`, how many elements can it store?
   (a) 5 (b) 25
   (c) 10 (d) 0
6. In linear search, when `VAL` is equal to the first element of the array, which case is it?
   (a) Worst case (b) Average Case
   (c) Best Case (d) Amortized case

Review Questions

1. Why do we need arrays?
2. What does an array name signify?
3. How is an array represented in memory?
4. How is a two-dimensional array represented in memory?
5. How can arrays be used for inter-function communication?
6. How are multi-dimensional arrays useful?
7. What happens when an array is initialized with fewer initializers as compared to its size?
8. What happens when an array is initialized with more initializers as compared to its size?
9. Why does storing of sparse matrices need extra consideration? How are sparse matrices stored efficiently in the computer's memory?
10. For an array declared as `int arr[50]`, calculate the address of `arr[35]`, if `Base(arr) = 1000` and `w = 2`.
11. Consider a 10 × 5 two dimensional array `Marks` which has base address = 2000 and the number of words per memory location of the array = 2. Compute the address of the element

- `Marks[8, 5]`, assuming that the elements are stored in row major order.

12. Given an array, `int arr[] = {9, 18, 27, 36, 45, 54, 63, 72, 81, 90, 99}`; trace the steps of binary search algorithm to find value 90 and 17 from the array.
13. Which technique of searching an element in the array do you prefer to use in specific situations?
14. Explain the concept of array of function pointers with the help of a program.
15. How can a one-dimensional array be used with functions?

Programming Exercises

1. Write a program which deletes all duplicate elements from the array.
2. Write a program that tests the equality of two one-dimensional arrays.
3. Write a program that reads an array of 100 integers. Display all pairs of elements whose sum is 50.
4. Write a program to input an array of 10 numbers and swap the value of the largest and smallest number.
5. Write a program to interchange second element with the second last element.
6. Write a program that calculates the sum of squares of the elements.
7. Write a program to calculate the number of duplicate entries in the array.
8. Write a program to arrange the values of an array in such a way that even numbers preceed the odd numbers.
9. Given a sorted array of integers, write a program to calculate the sum, mean, variance, and standard deviation of the numbers in the array.
10. Write a program to compute the sum and mean of the elements of a two-dimensional array.
11. Write a program that computes the sum of elements that are stored on the main diagonal of a matrix using pointers.
12. Write a program to count the total number of non-zero elements in a two-dimensional array.
13. Write a program to read and display an array of 10 floating point numbers using functions.
14. Write a program to read an array of 10 floating point numbers. Display the mean of these numbers till two decimal places.
15. Write a program to read an array of 10 floating point numbers. Display the smallest number and its position in the array.
16. Write a program to input the elements of a two-dimensional array. Then from this array make two arrays: one that stores all odd elements of the two-dimensional array and the other stores all even elements of the array.
17. Write a program to merge two integer arrays. Also display the merged array in reverse order
18. Write a program to transpose a $3 \times 3 \times 3$ matrix.
19. Write a program to multiply two $m \times n$ matrices.
20. Write a menu driven program to add, subtract, and transpose two matrices.
21. Write a program that reads a matrix and displays the sum of its diagonal elements.
22. Write a program that reads a matrix and displays the sum of the elements above the main diagonal.
    Hint: Calculate sum of elements of array A, where for an element A_{ij}, $i < j$
23. Write a program that reads a matrix and displays the sum of the elements below the main diagonal.
    Hint: Calculate sum of elementsof array A ,where for an element A_{ij}, $i > j$
24. Write a program that reads a square matrix of size $n \times n$. Write a function int is Upper Triangular(int a[][], int n) that returns 1 if the matrix is upper triangular.
    Hint: Array A is upper triangular if $A_{ij} = 0$ *for* $i > j$
25. Write a program that reads a square matrix of size $n \times n$. Write a function int is Lower Triangular(int a[][], int n) that returns 1 if the matrix is lower triangular.
    Hint: Array A is lower triangular if $A_{ij} = 0$ *for* $i < j$
26. Write a program that reads a square matrix of size $n \times n$. Write a function int is Symmetric (int a[][], int n) that returns 1 if the matrix is upper triangular.
    Hint: Array A is symmetric if $A_{ij} = A_{ij}$ *for all values of i and j*
27. Write a program to calculate XA + YB where A and B are matrices and X = 2, and Y = 3.
28. Write a program to read an array of 10 floating point numbers. Display the position of the second largest number.
29. Write a program to enter five single digit numbers in an array. Form a number using the array elements.
30. Write a program to find whether number 3 is present in the array arr[] = {1,2,3,4,5,6,7,8}.
31. Write a program to read a sorted floating point array. Update the array to insert a new number.
32. Write a program to read a floating point array. Update the array to delete the number from the specified location.

33. Modify the linear search program so that it operates on a sorted array.
34. Write a program to build an array of 100 random numbers in the range 1 to 100. Perform the following operations on the array.
    (a) Count the number of elements that are completely divisible by 3.
    (b) Display the elements of the array by displaying a maximum of ten elements in one line.
    (c) Display only the even elements of the array by displaying a maximum of ten elements in one line.
    (d) Count the number of odd elements.
    (e) Find the smallest element in the array.
    (f) Find the position of the largest value in the array.
35. Write a program to read two floating point arrays. Merge these arrays and display the resultant array.
36. Write a program to read two sorted floating point arrays. Merge these arrays and display the resultant array.
37. Write a program to read and display a p*q*r array.
38. Write a program to initialize all diagonal elements of a two-dimensional array to zero.
39. Consider a 10 × 10 two-dimensional array which has base address = 1000 and the number of memory locations per element of the array = 2. Now compute the address of the element arr[8][5] assuming that the elements are stored in row major order. Then calculate the same assuming the elements are stored in column major order.
40. Consider an array MARKS[20][5] which stores the marks obtained by 20 students in 5 subjects. Now write a program to
    (a) find the average marks obtained in each subject
    (b) find the average marks obtained by every student
    (c) find the number of students who have scored below **50** in their average
    (d) display the scores obtained by every student

Find the errors (if any) in the following declaration statements

- `int marks(10);`
- `int marks[10, 5];`
- `int marks[10],[5];`
- `int marks[10];`
- `int marks[ ];`
- `int marks[10] [5];`
- `int marks[9+1][6-1];`

Find the errors (if any) in the following initialization statements

- `int marks[ ] = {0,0,0,0};`
- `int marks[2][3] = {10,20,30,40};`
- `int marks[2,3] = {10, 20,30},{40,50,60};`
- `int marks[10] = {0};`

Find the output of the following codes

1.
```
int marks[],n,i,
for(i = 0; i < n; i++)
cin>>marks[i];
```

2.
```
int i, mark[];
for(i = 0; i < 10; i++)
cin>>marks[i];
```

3.
```
int marks[2][3], i, j;
for(i = 0; i < 2; i++)
for(j = 0; j<3;j++)
cin>>marks[i][j];
```

4.
```
using namespace std;
#include<iostream>
main()
{   int i, arr[10];
for(i = 0;i < 10;i++)
   arr[i*2] = 1;
for(i = 0;i < 10;i++)
   arr[i*2+1] = -1;
for(i = 0;i < 10;i++)
   cout<<"\t "<<arr[i];
}
```

5.
```
using namespace std;
#include<iostream>
main()
{   int arr[] = {0,1,2,0,1,2,0,1,2};
   cout<<"\n"<< arr[3];
   cout<<"\n"<< arr[arr[3]];
   cout<<"\n"<< arr[arr[3] + arr[1]];
   cout<<"\n"<< arr[arr[arr[arr[1]]]];
}
```

6.
```
using namespace std;
#include<iostream>
main()
{  int arr1[] = {0,1,2,0,1,2,0,1,2,0};
   int i, arr2[10];
   for(i = 0;i < 10;i++)
      arr2[i] = arr1[9-i];
   for(i = 0;i < 10;i++)
      cout<<"\t"<< arr2[i];
}
```

Fill in the Blanks

1. collection of elements
2. subscript (index)
3. whole number (integer)
4. contiguous
5. n
6. pointer
7. data type, name, and size
8. the element's position
9. Indexing
10. Base address
11. elements
12. Traversing
13. sentinel
14. continuous
15. constant pointer
16. array of arrays
17. 4th
18. end

True / False

1. True
2. False
3. True
4. True
5. False
6. False
7. True
8. False
9. True
10. True
11. False
12. True
13. True
14. True
15. False
16. False
17. True
18. True
19. False
20. True
21. True
22. True
23. True
24. False

Multiple Choice Questions (MCQ)

1. (b) 30
2. (d) 7
3. (b) 100
4. (b) 50
5. (c) 10
6. (c) Best Case

1. Why do we need arrays?

Answer: Arrays are needed to store multiple elements of the same data type in contiguous memory locations. They allow efficient access using indexing and help manage large datasets with ease, reducing code complexity compared to using multiple individual variables.

2. What does an array name signify?

Answer: An array name represents the base address of the array. It acts as a pointer to the first element and can be used to access elements using indexing.

3. How is an array represented in memory?

Answer: An array is represented in memory as a sequence of contiguous memory locations, each storing an element of the array. The base address gives the location of the first element, and subsequent elements are stored at regular offsets based on their data type size.

4. How is a two-dimensional array represented in memory?

Answer: A two-dimensional array is represented in row-major or column-major order in memory. In row-major order, elements of each row are stored sequentially, while in column-major order, elements of each column are stored sequentially.

5. How can arrays be used for inter-function communication?

Answer: Arrays can be passed as arguments to functions, allowing data to be shared and modified across different parts of a program without using global variables. They are typically passed by reference, avoiding unnecessary memory usage.

6. How are multi-dimensional arrays useful?

Answer: Multi-dimensional arrays allow the representation of complex data structures like matrices, tables, and graphs. They are used in applications such as image processing, game development, and scientific computing.

7. What happens when an array is initialized with fewer initializers than its size?

Answer: The remaining elements are automatically initialized to zero (for numeric data types) or NULL (for pointers).

8. What happens when an array is initialized with more initializers than its size?

Answer: A compilation error occurs because the number of initializers exceeds the defined array size.

9. Why does storing sparse matrices need consideration? How are sparse matrices stored efficiently in the computer's memory?

Answer: Sparse matrices have a large number of zero elements, and storing them as a regular array wastes memory. Instead, they are stored using efficient data structures like linked lists, compressed row storage (CRS), or dictionary-of-keys (DOK) to reduce memory usage.

10. For an array declared as `int arr[50];`, calculate the address of `arr[35]` if `base(arr) = 1000` and `w = 4`.

Answer: The address is calculated as:

$$\text{Address} = \text{Base Address} + (\text{Index} \times \text{Size of Each Element})$$

$$= 1000 + (35 \times 4) = 1000 + 140 = 1140$$

Thus, the address of `arr[35]` is **1140**.

11. Consider a `10 × 5` two-dimensional array `Marks` which has a base address = 2000 and the number of words per memory location of the array = 2. Compute the address of the element `Marks[2][2]`.

Answer: Using row-major order formula:

$$\text{Address} = \text{Base Address} + [(\text{Row Index} \times \text{Total Columns}) + \text{Column Index}] \times \text{Size of Element}$$

$$= 2000 + [(2 \times 5) + 2] \times 2$$

$$= 2000 + [10 + 2] \times 2$$

$$= 2000 + 12 \times 2$$

$$= 2000 + 24 = 2024$$

So, the address of `Marks[2][2]` is **2024**.

Q12. Trace the steps of the binary search algorithm to find values 90 and 17 from the given array.

Given Array (Sorted Order):

`arr[] = {9, 18, 27, 36, 45, 54, 63, 72, 81, 90, 99}`

Finding 90 using Binary Search:

1. Start with `low = 0`, `high = 10`, and `mid = (0+10)/2 = 5` (index 5 → value **54**).
2. Since `90 > 54`, update `low = mid + 1 = 6`.
3. New `mid = (6+10)/2 = 8` (index 8 → value **81**).
4. Since `90 > 81`, update `low = mid + 1 = 9`.
5. New `mid = (9+10)/2 = 9` (index 9 → value **90**).
6. Found **90** at index **9**.

Finding 17 using Binary Search:

1. Start with `low = 0`, `high = 10`, and `mid = (0+10)/2 = 5` (index 5 → value **54**).
2. Since `17 < 54`, update `high = mid - 1 = 4`.
3. New `mid = (0+4)/2 = 2` (index 2 → value **27**).
4. Since `17 < 27`, update `high = mid - 1 = 1`.
5. New `mid = (0+1)/2 = 0` (index 0 → value **9**).
6. Since `17 > 9`, update `low = mid + 1 = 1`.
7. New `mid = (1+1)/2 = 1` (index 1 → value **18**).
8. Since `17 < 18`, update `high = mid - 1 = 0`.
9. `low > high`, so **17 is not found in the array**.

Q13. Which technique of searching an element in the array do you prefer to use in specific situations?

- **Binary Search:** Used when the array is **sorted** because it has a time complexity of **O(log n)**.
- **Linear Search:** Used for **unsorted** or **small arrays** where sorting is not required. It has a time complexity of **O(n)**.
- **Hashing:** Used for **fast lookups** when space is not an issue, providing average time complexity **O(1)**.
- **Interpolation Search:** Used when elements are uniformly distributed for better performance than Binary Search.

Q14. Explain the concept of an array of function pointers with the help of a program.

An **array of function pointers** stores multiple function addresses and allows calling different functions dynamically.

Example Code in C:

c Copy Edit

```c
#include <stdio.h>

void add(int a, int b) { printf("Sum = %d\n", a + b); }
void subtract(int a, int b) { printf("Difference = %d\n", a - b); }
void multiply(int a, int b) { printf("Product = %d\n", a * b); }

int main() {
    void (*funcArr[3])(int, int) = {add, subtract, multiply};

    funcArr[0](10, 5);  // Calls add function
    funcArr[1](10, 5);  // Calls subtract function
    funcArr[2](10, 5);  // Calls multiply function

    return 0;
}
```

Output:

ini Copy Edit

```ini
Sum = 15
Difference = 5
Product = 50
```

This allows switching between functions dynamically.

Q15. How can a one-dimensional array be used with functions?

A **1D array** can be passed to a function in three ways:

1. Passing by Reference (Using Pointers)

c Copy Edit

```c
void printArray(int *arr, int size) {
    for (int i = 0; i < size; i++)
        printf("%d ", arr[i]);
    printf("\n");
}

int main() {
    int numbers[] = {10, 20, 30, 40};
    printArray(numbers, 4);
    return 0;
}
```

Output:

```
10 20 30 40
```

2. Passing by Address

c Copy Edit

```c
void modifyArray(int arr[], int size) {
    arr[0] = 99;  // Modifies original array
}

int main() {
    int nums[] = {1, 2, 3};
    modifyArray(nums, 3);
    printf("%d\n", nums[0]); // Output: 99
    return 0;
}
```

3. Returning an Array from a Function

c Copy Edit

```c
int* getArray() {
    static int arr[3] = {5, 10, 15};
    return arr;
}

int main() {
    int *p = getArray();
    printf("%d\n", p[1]); // Output: 10
    return 0;
}
```

This approach requires `static` storage to avoid returning local memory.