CHAPTER

7

Arrays and Subscripted Variables

CHAPTER OBJECTIVES

- One-dimensional Array
- Two-dimensional Array
- Array Declaration

A list of related values is stored in RAM in the form of an array. Values in an array are identified using array name with subscripts. Single subscripted variable is referred to a one-dimensional or linear array; two subscripted variable is referred to a two-dimensional or matrix array; and so on.

ONE-DIMENSIONAL ARRAY

Values in a mathematical set are written as shown below.

$x = \{3.1, 3.2, 4.6, 2.9, 3.8, 3.2\}$

These values are referred in mathematics as follows.

x_0, x_1, x_2 and so on

In C language they are represented as follows.

`x[0] x[1] x[2]` and so on

It may be imagined that these values are stored in RAM as follows.

Subscript →	0	1	2	3	4	5
Array name → x	3.1	3.2	4.6	2.9	3.8	3.2

TWO-DIMENSIONAL ARRAY

Consider a matrix of size 3 × 4:

$$A = \begin{pmatrix} 2 & 4 & 3 & 0 \\ 3 & 8 & -1 & 6 \\ -2 & 0 & -5 & 6 \end{pmatrix}$$

These values are referred in mathematics as follows.

$A_{0,0}$ $A_{0,1}$ $A_{0,2}$ and so on

In C language they are represented as follows.

`a[0][0] a[0][1] a[0][2]` and so on

It may be imagined that these values are stored in RAM as follows.

Array name → a

	0	1	2	3
0	2	4	3	0
1	3	8	−1	6
2	−2	0	−5	6

← Column subscripts

↑ Row subscripts

ARRAY DECLARATION

Arrays are declared using type declaration statements with a maximum number of values accommodated in them. Consider the following example

```
int x[50], a[3][4];
```

This statement declares an one-dimensional array x having 50 elements and a two-dimensional array a with 12 elements (3 rows × 4 columns = 12). An element means a storage location in an array.

Note:

1. Array size to store values is always an integer; a declaration like `int x[n];` is not permitted.
2. Subscript value beyond the array size should not be used. This will not be detected by the compiler and it leads to error during program execution. For example, `x[51]` should not be used.

Accessing Values in an Array

Values in an array are accessed using array name with subscripts in bracket [ ].

Normally, the list of values to the array is read using a **for** loop. Consider the following example:

```
for(i = 0;  i < 6; i++)
   scanf("%d", &x[i]);
```

When this loop is executed, the value of `i` varies from 0 to 5 in steps of 1, and **scanf()** is executed every time to read a value of a one-dimensional array as follows.

First time: i = 0

Read value for `x[0]`. e.g. enter 3.1

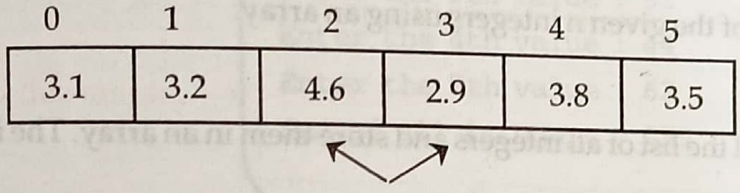
Second time: i = 1

Read value for `x[1]`. e.g. enter 3.2

Finally: i = 5

Read value for `x[5]`. e.g. enter 3.5

Thus the 6 values in the list are read and stored in RAM as illustrated below.



Memory cells in RAM

RAM consists of many memory cells, each of size 1 byte. Each element is allotted 2 bytes since the size of an integer is 2 bytes. The size of this array is 12 bytes (2 × 6 = 12).

Note that **for** loops can also be used to print and process the values stored in the array.

Initializing Values in an Array

Values of an array may be initialized while declaring them. Consider the following example.

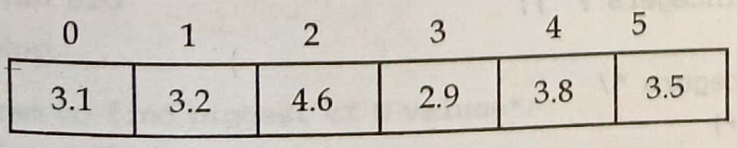
```
float x[6] = {3.1, 3.2, 4.6, 2.9, 3.8, 3.5 };
```

or

```
float x[] = {3.1, 3.2, 4.6, 2.9, 3.8, 3.5 };
```

Note that the array size is ignored and is considered as `x[6]` automatically as per the number of values available.

The values assigned are stored as follows.




```
x[0] = 3.1
x[1] = 3.2
```

```
x[2] = 4.6
x[3] = 2.9
x[4] = 3.8
x[5] = 3.5
```

Similarly an unit matrix of size 3 × 3 may be initialized as follows.

```
int u[3][3] = {{1,0,0}, {0,1,0}, {0,0,1}};
```

The values are stored in the form of an unit matrix as shown below.

```
a[0][0] = 1
a[0][1] = 0
a[0][2] = 0
.
.
.
a[2][2] = 1
```

u	0	1	2
1	1	0	0
1	0	1	0
2	0	0	1

Example 1

Write a C program to find the sum of the given n integers using an array.

Solution

Consider the following steps to read the list of all integers and store them in an array. The integers a then added to get the sum.

Step 1: Read n

Step 2: Loop to read n values of the list.

Step 3: SUM ← 0

Step 4: Loop to add all values to get SUM of those values

Step 5: Print SUM

Step 6: Stop

```
/* Program to find sum of N integers */
#include <stdio.h>
#include <conio.h>
main()
{
    int x[50],n,i,sum;
    clrscr();
    printf("\n How many integers ? ");
    scanf("%d",&n);
    /* loop to read N integers */
    for(i = 0; i < n; i++)
    {
```

```
        printf("\n Enter the %dth value : ",i+1);
        scanf("%d",&x[i]);
    }
    sum = 0;
    /* loop to find sum of all integers */
    for(i = 0; i < n; i++)
        sum = sum + x[i];
    printf("\n Sum of all integers = %d",sum);
    getch();
}
```

When this program is executed, the user has to enter the values of integers which are read using **for** loop. The second **for** loop is written to add all values. The result is printed as follows.

```
How many integers ? 5
Enter the 1th value : 36
Enter the 2th value : 45
Enter the 3th value : 52
Enter the 4th value : 44
Enter the 5th value : 62
Sum of all integers = 239
```

Note: Read 1st, 2nd and 3rd for 1th, 2th and 3th in the messages.

Example 2

Write a C program to find the biggest of given n numbers.

Solution

Consider a **for** loop to read the list of n numbers and store them in an array. The first number in the lis is assumed as `big` and is compared with the remaining numbers to get the biggest number. Steps i this program are written as follows.

Step 1 : Read n

Step 2 : Loop to read n numbers of the list

Step 3 : `BIG = x[0]`

Step 4 : Loop to find the biggest by comparing it with the remaining numbers

Step 5 : Print `BIG`

Step 6 : Stop

```
/*Program to find biggest of N values*/
#include <stdio.h>
#include <conio.h>
```

```
main()
{   int x[50],n,i,big;
    clrscr();
    printf("\n How many numbers ? ");
    scanf("%d",&n);
    printf("\n Enter all those numbers \n");
    for(i = 0; i < n; i++)
       scanf("%d",&x[i]);
    /* assume first value as big */
    big = x[0];
    /* loop to find the biggest by comparing from second number onwards */
    for(i = 1; i < n; i++)
       if(x[i] > big)
          big = x[i];
    printf("\n %d is the biggest number",big);
    getch();
}
```

When this program is executed, the user has to enter the number of values available and the values are read using **for** loop. The second **for** loop is used to find the biggest value. Finally, the biggest number is displayed as follows.

```
How many numbers ? 5
Enter all those numbers
25  -228  0  185  36
185 is the biggest number
```

Example 3

Write a C program to find the arithmetic mean, variance and standard deviation of any n values.

Solution

An array may be used to store the list of 'n' values. The arithmetic mean, variance and standard deviation are obtained using the following:

$$\text{Mean } \bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

$$\text{Variance } \sigma_x = \frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n}$$

$$\text{Standard deviation} = \sqrt{\sigma_x}$$

Following steps are used to find the mean, variance and standard deviation.

Step 1: Read n

Step 2 : Loop to read n values in the list

Step 3 : SUM = 0

Step 4 : Loop to find SUM of all values

Step 5 : $\bar{x} = \frac{SUM}{n}$

Step 6 : VSUM = 0

Step 7 : Loop to find the numerator VSUM to find variance

Step 8 : $\sigma_x = \frac{VSUM}{n}$

Step 9 : Standard deviation = $\sqrt{\sigma_x}$

Step 10 : Print $\bar{x}$, σ_x, standard deviation

Step 11 : Stop

```
/* Program to find Mean,Variance and Standard Deviation */
#include <stdio.h>
#include <conio.h>
#include <math.h>
main()
{   float x[50],sum,vsum,xbar,sigmax,sd;
    int n,i;
    clrscr();
    printf("\n How many values ? ");
    scanf("%d",&n);
    printf("\n Enter all values in the list \n");
    for(i = 0; i < n; i++)
        scanf("%f",&x[i]);
    sum = 0;
    /* loop to find sum of all values */
    for(i = 0; i < n; i++)
        sum = sum + x[i];
    xbar = sum/n;
    vsum = 0;
    /* loop to find the numerator vsum to find variance */
    for(i = 0; i < n; i++)
        vsum = vsum + (x[i] - xbar)*(x[i] - xbar);
    sigmax = vsum/n;
    sd = sqrt(sigmax);
    printf("\n Arithmetic mean = %0.3f",xbar);
```

```
    printf("\n Variance = %0.3f",sigmax);
    printf("\n Standard deviation = %0.3f",sd);
    getch();
}
```

When this program is executed, the user has to enter the number of n and their values. Consi the following list of values to run the program.

X = {3.1, 3.8, 3.6, 4.0, 3.4, 3.8}

The arithmetic mean, variance, standard deviation are obtained and printed as follows.

```
How many values ? 6
Enter all values in the list
3.1  3.8  3.6  4.0  3.4  3.8
Arithmetic mean = 3.617
Variance = 0.088
Standard deviation = 0.297
```

Example 4

Write a C program to read a list of test marks (integers in the range 0—100) of 50 students. Calcula the mean of marks and print a list of marks greater than the mean (rounded to 2 decimals).

Solution

The steps followed in the previous problem can be used to find the mean. Use another loop to print th list of values which are greater than the mean value. Note that the mean value is a float type numbe

```
/* Program to find mean of marks */
#include <stdio.h>
#include <conio.h>
#include <math.h>
main()
{   int x[50],n,i,sum;
    float mean;
    clrscr();
    printf("\n How many students ? ");
    scanf("%d",&n);
    printf("\n Enter all the marks \n");
    for(i = 0; i < n; i++)
       scanf("%d",&x[i]);
    sum = 0;
```

The difference in magnitude of the numbers is obtained as follows.

diff = | psum - | nsum | |
= | 25 - | - 37 | |
= | 25 - 37 |
= | - 12 |
= 12

The C program to find the sum of positive and negative numbers and to print the differe magnitude between them is written as follows.

```
/* Program to find sum of +ve and -ve values */
#include <stdio.h>
#include <conio.h>
#include <math.h>
main()
{
    float x[50],psum,nsum,diff;
    int n,i;
    clrscr();
    printf("\n How many values ? ");
    scanf("%d",&n);
    printf("\n Enter all values in the list \n");
    for(i = 0; i < n; i++)
        scanf("%f",&x[i]);
    psum = 0;
    nsum = 0;
    /* loop to find sum of all positive and negative values */
    for(i = 0; i < n; i++)
    {
        if(x[i] > 0)
            psum = psum + x[i];
        else
            nsum = nsum + x[i];
    }
    printf("\n Sum of positive values = %0.2f",psum);
    printf("\n Sum of negative values = %0.2f",nsum);
    if(psum > fabs(nsum))
        printf("\n Positive sum is greater in magnitude");
    else
        printf("\n Negative sum is greater in magnitude");
    diff = fabs(psum - fabs(nsum));
    printf("\n Difference in magnitude = %0.2f",diff);
    getch();
}
```

When this program is executed, the sample values are entered and the results are obtained follows.

```
How many values ? 6
Enter all values in the list
8 -12 -16 12 -9 5
Sum of positive values = 25.00
Sum of negative values = -37.00
Negative sum is greater in magnitude
Difference in magnitude = 12.00
```

Example 6

Write a C program to sort a set of n numbers in ascending order and explain the algorithm used.

Solution

There are many methods available to sort (arrange) numbers in ascending/descending order. Selection sorting algorithm is one of the simplest algorithm used to sort numbers.

Consider a list of values:

6 -2 8 3 13 10

Compare the first number with the second, then with the third and so on till the smallest value moves to the first place.

-2 6 8 3 13 10

Compare the second number with the third, then with the fourth and so on till the second smallest value moves to the second place.

-2 3 8 6 13 10

The procedure is repeated to move the third smallest to the third place and so on. Finally the numbers are arranged as follows.

-2 3 6 8 10 13

The steps involved in arranging the numbers in ascending order are given below.

Step1 : Read n

Step2 : Loop to read n values of the list

Step 3: Loop to arrange the numbers by comparison

if $(x_i > x_{i+1})$ then

exchange the values

Step 4: Loop to print the arranged list

Step 5: Stop

```
/* Program to sort numbers in ascending order */
#include <stdio.h>
#include <conio.h>
main()
{   int x[50],n,i,j,temp;
    clrscr();
    printf(" How many numbers ? ");
    scanf ("%d",&n);
    printf(" \n Enter the list of %d numbers \n",n);
    for(i = 0; i < n; i++)
       scanf("%d",&x[i]);
    /* loop to arrange the numbers in ascending order */
    for(i = 0; i < n-1; i++)
       for(j = i+1; j < n; j++)
          if(x[i] > x[j])
          {
             temp = x[i];
             x[i] = x[j];
             x[j] = temp;
          }
    printf("\n Numbers in ascending order \n");
    for(i = 0; i < n; i++)
       printf("%5d",x[i]);
    getch();
}
```

When this program is executed, the user has to enter the number of n and their values. Whe value of a number in a preceding position is greater than the one in a succeeding position, the v are exchanged using the statements

```
temp = x[i];
x[i] = x[j];
x[j] = temp;
```

The values are obtained in the ascending order and are printed as follows.

```
How many numbers ? 4
Enter the list of 4 numbers
32  -10   20    5
Numbers in ascending order
-10   5   20   32
```

Example 7

Write a C program to search the key value in a set of n values. Print the position of the key value if it is a successful search.

Solution

There are many methods available to search a key value in list of 'n' values. Linear search algorithm is the simplest method to search the given key value in a list.

Consider a list of values:

6 -2 8 3 13 10

Let the key value to be searched is 3.

Compare the key value with the first value in the list, then with the second value and so on.

When a value in the list is equal to the given key value, stop searching and display the position.

The steps followed to search a key value in a list of n values are given below.

Step 1: Read n

Step 2: Loop to read n values in the list

Step 3: Read S / * S is the key value to be searched in the list * /

Step 4: Loop to compare S with the values in the list

```
if (S = x [ i ]) then
        print location
stop
```

Step 5: Print "The key value not in the list"

Step 6: Stop

```
/* Program to search key value in a list */
#include <stdio.h>
#include <conio.h>
main()
{
int x[50],n,i,s;
clrscr();
printf("\n How many values in the list ? ");
scanf("%d",&n);
printf("\n Enter all values in the list \n");
for(i = 0; i < n; i++)
   scanf("%d",&x[i]);
printf("\n Enter the key value to be searched : ");
scanf("%d",&s);
```

```
/* loop to search the key value in the list */
for(i = 0; i < n; i++)
    if(s == x[i])
    {
        printf("\n %d is available in %dth location",s,i+1);
        getch();
        exit(0);
    }
printf("\n The key value %d is not present in the list",s);
getch();
}
```

When this program is executed, the user has to first enter the list of values and then the key value to be searched in the list. The given key value is compared with the list of values starting from position one. When a value in the list is equal to the key value, the value will be printed along with the location in which it is present.

If the value is not present in the list of values then the message "The key value is not in the list" will be displayed.

```
How many values in the list ? 6
Enter all values in the list
6  -2  8  3  13  10
Enter the key value to be searched : 3
3 is available in 4th location
```

Note that the function **exit(0)** is used to stop the program execution. This may be achieved indirectly using a `flag` variable as follows.

```
flag = 0;
for(i = 0; i< n && flag == 0; i++)
if(s == x[i])
    flag = 1;
if(flag == 1)
    printf(" \n %d is available in %dth location ",s,i);
else
    printf("\n The key value %d is not present in the list", s);
```

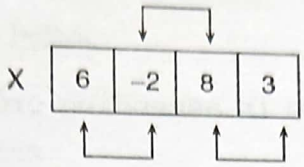
This will also produce the same result.

Example 8

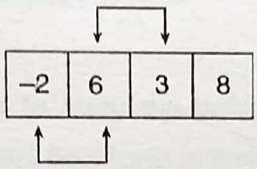
Write a C program to sort n given numbers using bubble sort and also find the number of exchanges and passes.

Solution

The bubble sort technique is used to arrange the given set of numbers in ascending order. Consider a list of 'n' values.

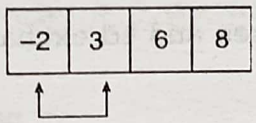


Compare the values in the first and second position, second and third position, third and fourth position, and so on. If `x[0] > x[1]` then exchange the values. This process is continued till the largest value is moved to the last position of the array.



The comparison is continued up to (`n-1`) position and the values are exchanged (if necessary). Note that the second largest element is moved to the second last position of the array.

The comparison is repeated again up to (`n-2`) position and the values are exchanged (if necessary). Now the third largest element is moved to the third last position.



The following steps are written to sort the numbers.

Step 1: Read n

Step 2: Loop to read n numbers

Step 3: Loop to repeat the procedure (n-1) times

```
begin
    loop to compare the consecutive elements
        if( x[ j ] > x[j+1] ) then
            exchange the values
end
```

Step 4: Loop to print the sorted list

Step 5: Stop

```
/* Program to sort numbers using bubble sort technique */
#include <stdio.h>
#include <conio.h>
```

```
main()
{
    int x[10],temp,i,j,n,exchng = 0;
    clrscr();
    printf("\n How many numbers ? ");
    scanf("%d",&n);
    printf("\n Enter all numbers in the list \n");
    for(i = 0; i < n; i++)
       scanf("%d",&x[i]);
    /* loop to sort the numbers in ascending order */
    for(i = 0; i < n-1; i++)
       for(j = 0; j <= n-i-1; j++)
       if(x[j] > x[j+1])
       {
          temp = x[j];
          x[j] = x[j+1];
          x[j+1] = temp;
          exchng++;
       }
       printf("\n The sorted list is \n");
    /* loop to print the sorted list */
    for(i = 0; i < n; i++)
       printf("%5d",x[i]);
    printf("\n Sorted in %d passes and %d exchanges",n-1,exchng);
    getch();
}
```

When this program is executed, the user has to enter the number of 'n' and their values. A nested **for** loop is used to compare and rearrange the values in ascending order.

```
How many numbers? 4
Enter all numbers in the list
6  -2  8  3
The sorted list is
-2  3  6  8
Sorted in 3 passes and 3 exchanges
```

Note that bubble sort is faster than the selection sort technique.

Example 9

Write a C program to add two matrices of order m × n.

Solution

Consider two matrices A and B of order 2 × 2 and add them.

$$\begin{pmatrix} 2 & -2 \\ 0 & 4 \end{pmatrix} + \begin{pmatrix} 6 & 2 \\ 4 & -5 \end{pmatrix} = \begin{pmatrix} 8 & \\ 4 & \end{pmatrix}$$

The steps to be followed are given below.

Step 1: Read m, n

Step 2: Loop to read A matrix values

Step 3: Loop to read B matrix values

Step 4: Loop to add the matrices to find C matrix

Step 5: Loop to print the resultant C matrix values

Step 6: Stop

Note that the loops are to be written to vary the values of row and column subscripts in the matrices. For example, the following nested **for** loop is written to read values of A matrix.

```
for(i = 0; i < m; i++)
   for(j = 0; j < n; j++)
      scanf("%d", &a[i][j]);
```

The outer loop index variable `i` is used to refer the row subscript and inner loop index variable `j` is used to refer the column subscript. When `i = 0`, the column subscript `j` varies from `0` to `n` to read values row wise.

```
/* Program for matrix addition */
#include <stdio.h>
#include <conio.h>
main()
{ int a[10][10],b[10][10],c[10][10],m,n,i,j;
  clrscr();
  printf("\n How many rows and columns ? ");
  scanf("%d %d",&m,&n);
 /* loop to read values of A matrix */
  printf(" Enter A matrix \n");
  for(i = 0; i < m; i++)
    for(j = 0; j < n; j++)
     scanf("%d",&a[i][j]);
 /* loop to read values of B matrix */
  printf(" Enter B matrix \n");
  for(i = 0; i < m; i++)
    for(j = 0; j < n; j++)
```

```
        scanf("%d",&b[i][j]);
/* loop to add two matrices */
  for(i = 0; i < m; i++)
    for(j = 0; j < n; j++)
      c[i][j] = a[i][j] + b[i][j];
/* loop to print resultant matrix */
  printf("\n Resultant matrix is \n");
  for(i = 0; i < m; i++)
    {
     for(j = 0; j < n; j++)
       printf("%6d",c[i][j]);
       printf("\n");
    }
  getch();
}
```

When this program is executed, the user has to enter the order (m, n) and the values of the matrice The resultant matrix is displayed below.

```
How many rows and columns ? 2  2
Enter A matrix
2  -2
0   4
Enter B matrix
6   2
4  -5
Resultant matrix is
8    0
4   -1
```

Note that `printf("\n");` will print the values row wise. Thus the resultant matrix gets displayed i matrix form.

Example 10

Write a C program to multiply A matrix of order m × n with B matrix of order n × l.

Solution

Consider two matrices A and B of order 2 × 2. Multiply them to get the resultant matrix C. Note tha $A_{m \times n} \times B_{n \times l}$ produces the resultant matrix $C_{m \times l}$.

$$\begin{pmatrix} 2 & -2 \\ 0 & 4 \end{pmatrix} \times \begin{pmatrix} 6 & 2 \\ 4 & -5 \end{pmatrix} = \begin{pmatrix} 4 & 14 \\ 16 & -20 \end{pmatrix}$$

The steps followed to multiply matrices are similar to those explained in addition of matrices except the multiplication method.

```
/* Program for matrix multiplication */
#include <stdio.h>
#include <conio.h>
#include <math.h>
main()
{
  int a[10][10],b[10][10],c[10][10],m,n,i,j,l,k;
clrscr();
printf("\n Enter order of A matrix : ");
scanf("%d %d",&m,&n);
 /* loop to read values of A matrix */
printf(" Enter A matrix \n");
for(i = 0; i < m; i++)
    for(j = 0; j < n; j++)
       scanf("%d",&a[i][j]);
printf("\n Enter order of B matrix : ");
scanf("%d %d",&n,&l);
/* loop to read values of B matrix */
printf(" Enter B matrix \n");
for(i = 0; i < n; i++)
   for(j = 0; j < l; j++)
      scanf("%d",&b[i][j]);
/* loop to multiply two matrices */
for(i = 0; i < m; i++)
   for(j = 0; j < l; j++)
   {
      c[i][j] = 0;
      for(k = 0; k < n; k++)
         c[i][j] = c[i][j] + a[i][k]*b[k][j];
   }
/* loop to print resultant matrix */
printf("\n Resultant matrix is \n");
for(i = 0; i < m; i++)
{
   for(j = 0; j < l; j++)
      printf("%6d",c[i][j]);
   printf("\n");
}
getch();
```

When this program is executed, the user has to first enter the order (m, n) of A matrix and its values and then the order (n, l) of B matrix and its values. Note that the innermost loop

```
for(k = 0; k < n; k++)
    c[i][j] = c[i][j] + a[i][k] * b[k][j];
```

is used to multiply row elements of A matrix with respective column elements of B matrix and add the results to get an element for C matrix. This is repeated in the outer loops to get the other elements in the resultant matrix.

```
Enter order of  A matrix : 2 2
Enter A matrix
2    -2
0    4
Enter order of B matrix: 2 2
Enter B matrix
6    2
4    -5
Resultant matrix is
4    14
16   -20
```

Example 11

Write a C program to find and print the transpose of a given matrix.

Solution

Consider a matrix A of order m × n. Its transpose (A^T) is obtained by writing the first row of A matrix as the first column of A^T, the second row of A matrix as the second column of A^T and so on. The order of transpose (A^T) matrix is n × m.

$$A = \begin{pmatrix} -3 & 6 & 0 \\ 3 & 2 & 8 \end{pmatrix}_{2x3} \qquad A^T = \begin{pmatrix} -3 & 3 \\ 6 & 2 \\ 0 & 8 \end{pmatrix}_{3x2}$$

```
/* Program to transpose a given matrix */
#include <stdio.h>
#include <conio.h>
main()
{   int a[10][10],at[10][10],m,n,i,j;
    clrscr();
    printf("\n Enter order of the matrix : ");
    scanf("%d %d",&m,&n);
/* loop to read values of the  matrix */
    printf("\n Enter the matrix values \n");
    for(i = 0; i < m; i++)
        for(j = 0; j < n; j++)
            scanf("%d",&a[i][j]);
```

```
/* loop to Transpose the matrix */
   for(i = 0;i < m; i++)
      for(j = 0; j < n; j++)
         at[j][i] = a[i][j];
/* loop to print the resultant matrix */
   printf("\n The transposed matrix is \n");
   for(i = 0; i < n; i++)
   {
      for(j = 0; j < m; j++)
         printf("%6d",at[i][j]);
      printf("\n");
   }
   getch();
}
```

When this program is executed, the user has to enter the order (m, n) and values of the given matrix. The transpose matrix is printed as follows.

```
Enter order of the matrix : 2 3
Enter the matrix values
-3  6  0
 3  2  8
The transposed matrix is
-3    3
 6    2
 0    8
```

Example 12

Write a C program to determine whether the given matrix is symmetric or not.

Solution

Consider a square matrix A of order m × m. When the row elements and column elements are identical, the matrix is called as a symmetric matrix. For example,

$$A = \begin{pmatrix} 5 & 3 & 8 \\ 3 & 1 & -2 \\ 8 & -2 & 4 \end{pmatrix}$$

is a symmetric matrix.

```
/* Program to check for symmetric matrix */
#include <stdio.h>
#include <conio.h>
main()
```

```
{   int a[10][10],m,i,j,flag;
    clrscr();
    printf("\n Enter order of the square matrix : ");
    scanf("%d",&m);
/* loop to read values of A matrix */
    printf("\n Enter the matrix \n ");
    for(i = 0; i < m; i++)
       for(j = 0; j < m; j++)
          scanf("%d",&a[i][j]);
/* loop to check for symmetric matrix */
    flag = 0;
    for(i = 0; i < m && flag == 0; i++)
       for(j = 0; j < m && flag == 0; j++)
          if(a[i][j] == a[j][i])
             continue;
          else
             flag = 1;
    if(flag == 0)
       printf("\n The given matrix is a symmetric matrix");
    else
       printf("\n The given matrix is not a symmetric matrix");
    getch();
}
```

When this program is executed, the user to enter the order `m` and values of the given matrix. Note that symmetric matrix is possible only for a square matrix.

The first row elements are compared with the first column elements. If they are identical then the loop is continued to check the second row with second column and so on. When any value does not match, `flag` will be assigned to `1` which leads to the termination of the loop. The result will depend on the value of the `flag` variable.

```
Enter order of the square matrix :3
Enter the matrix
5   3   8
3   1  -7
8  -7   4
The given matrix is a symmetric matrix
```

Example 13

Write a C program to find the trace of a given square matrix of order m × m.

Solution

We know that the trace of a matrix is defined as the sum of the leading diagonal elements. Also note that trace is possible only for a square matrix. Consider the following example.

$$A = \begin{pmatrix} 3 & 2 & -1 \\ 4 & 1 & 8 \\ 6 & 4 & 2 \end{pmatrix}$$

Trace of A matrix = $A_{11} + A_{22} + A_{33} = 3 + 1 + 2 = 6$

Note that row `i` and column `j` are equal for a diagonal element.

```
/* Program to find trace of square matrix */
#include <stdio.h>
#include <conio.h>
main()
{   int a[10][10],m,i,j,sum;
    clrscr();
    printf("\n Enter order of the square matrix : ");
    scanf("%d",&m);
 /* loop to read values of A matrix */
    printf("\n Enter the matrix \n");
    for(i = 0; i < m; i++)
        for(j = 0;j < m; j++)
            scanf("%d",&a[i][j]);
  /* loop to find trace of the matrix */
    sum = 0;
    for(i = 0; i < m; i++)
        sum = sum + a[i][i];
    printf("\n Trace of the matrix = %d",sum);
    getch();
}
```

When this program is executed, the user has to enter the order `m` and values of the given matrix. A **`for`** loop is written to find the sum of the diagonal elements. Note that the index variable of the loop `i` is used for row and column subscripts to represent the diagonal elements.

Example 13

Write a C program to find the trace of a given square matrix of order m × m.

Solution

We know that the trace of a matrix is defined as the sum of the leading diagonal elements. Also note that trace is possible only for a square matrix. Consider the following example.

$$A = \begin{pmatrix} 3 & 2 & -1 \\ 4 & 1 & 8 \\ 6 & 4 & 2 \end{pmatrix}$$

Trace of A matrix = $A_{11} + A_{22} + A_{33} = 3 + 1 + 2 = 6$

Note that row `i` and column `j` are equal for a diagonal element.

```
/* Program to find trace of square matrix */
#include <stdio.h>
#include <conio.h>
main()
{  int a[10][10],m,i,j,sum;
   clrscr();
   printf("\n Enter order of the square matrix : ");
   scanf("%d",&m);
/* loop to read values of A matrix */
   printf("\n Enter the matrix \n");
   for(i = 0; i < m; i++)
      for(j = 0;j < m; j++)
         scanf("%d",&a[i][j]);
 /* loop to find trace of the matrix */
   sum = 0;
   for(i = 0; i < m; i++)
      sum = sum + a[i][i];
   printf("\n Trace of the matrix = %d",sum);
   getch();
}
```

When this program is executed, the user has to enter the order `m` and values of the given matrix. A `for` loop is written to find the sum of the diagonal elements. Note that the index variable of the loop `i` is used for row and column subscripts to represent the diagonal elements.

```
Enter order of the square matrix : 3
Enter the matrix
3  2 -1
4  1  8
6  4  2
Trace of the matrix = 6
```

REVIEW QUESTIONS AND EXERCISES

1. Define a one-dimensional array having 50 elements. Write a C program to calculate the sum of values stored in even numbered elements starting from the subscript 0.
2. Write a C program to find the sum of odd and even numbers in a given list of values.
3. Write a C program to copy all the elements of the linear array A into the corresponding position in the linear array B. There are N corresponding positions in the linear array B and there are N elements.
4. Write a C program to read n integers from keyboard and store them in the array AR. The odd elements in the AR are copied into array OAR and others are copied into array EAR. Print values in arrays OAR and EAR.
5. Write a C program to print the smallest value in a given array along with its position in the array.
6. Write a C program to find the second largest value in array of 'n' elements.
7. Write a C program to read $x_1, x_2, x_3 \cdots x_n$ of raw data and compute the number of data above and below the mean (average). Use `while` statement to find average.
8. Write a C program to read a list of n integers. Count the number of positive and negative integers. Also count the number of zeroes in the list and display the result.
9. Write a C program to find the maximum and minimum in an array of 100 integers and their locations.
10. Write a C program to sort the elements of an array in descending order.
11. An integer array NUMBER contains 100 elements. Write a C program to interchange the 1st and 100th elements, the 2nd and 99th elements and so on of the array NUMBER. The program should stop after interchanging the 50th and 51st elements.
12. Given are two linear arrays of integers, one containing 20 elements and the other containing 15 elements. After reading data to obtain values for all items in each array, print the values of only those integers that appear in both the arrays. Write a C program to implement this.
13. Write a C program to search an element in an array of n elements.

14. Write a C program to list all the numbers which are divisible by 2 in an array of 'n' integers.
15. Given are two one-dimensional arrays A and B which are sorted in ascending order. Write a program to merge them into a single sorted array C that contains every item from arrays A and B in ascending order.
16. Write a C program to add two m × n matrices and store the results in a third matrix.
17. Write a C program to find the product of two matrices.
18. Write a program to read a matrix (m × n) columnwise, then transpose the matrix and get the output printed rowwise in a matrix form.
19. Given a square matrix A of 10 × 10 size, write a C program to test if A is a diagonal matrix.
20. Given a matrix A of order 5 × 5, write a C program to find the row sum, column sum and diagonal sum of the values.
21. Explain how looping structures are used to read/write one, two and three dimensional arrays.
22. Given a matrix A of size 10 x 10, write a C program to find the sum of all positive and negative elements.
23. Write a program in C to read N × N matrix and compute its trace and norm. Print the matrix in the form of N rows and N columns.

    [The trace of a matrix is the sum of the leading diagonal elements. The norm of a matrix is the largest value of the row sum of the given matrix. The row sum is obtained as the absolute value of elements in each row.]
24. Four tests are given to a class of 30 students. Write a C program that calculates the average in each test and the class average of all tests.

SHORT QUESTIONS AND ANSWERS

1. Find the number of elements of the following arrays.

```
float f[4][8][2];
char  si[] = "ibm";
```

f array will have 64 elements (4 × 8 × 2 = 64)

si array will have 3 elements considering the 3 characters assigned in the declaration.

2. In C an array of characters is called _____.

string

3. Discuss the syntax of array in C.

An array is declared with specific number of elements in type declaration statement. It has the following form.

```
type name[n];
```

where type is the data type of values in the array.

name is the array name.

n is the maximum number of values that can be stored in the array.

Example:

```
int x[100], a[3][4];
```

This statement declares a one-dimensional array with 100 elements and a two-dimensional array with 12 elements (3 × 4 = 12).

4. How does C handle the values in an array internally?
The values in an array are stored in RAM continuously in adjacent locations. Actually these memory locations are reserved by the C compiler at the time of compilation. The values in an array are accessed using the array name with the subscripts in brackets. Normally the subscript values start from 0.

5. The array declaration for a list of 50 floating point temperature is _____.
```
float temp[50];
```

6. How is an array of 10 integers declared and used in C ?
An array of 10 integers is declared as follows.
```
int x[10];
```
The values of the array are accessed using the array name with one subscript in bracket. For example, `x[0] x[1]...x[9]`
Normally array values are read/written using a `for` loop. For example,
```
for( i = 0; i < 10; i++)
        scanf(" %d", &x[i]);
```

7. An array is a group of related data items that share a _____.
memory location in RAM

8. What are the conditions that must be satisfied by all the elements of any given array?
(i) The values stored in an array have the same data type as declared in the array. For example, `int x[10]` can be used to store only integers.
(ii) The number of values stored in an array is always less than or equal to the size of the array entered in the declaration statement. For example, `int x[50]` can store 50 integers.

9. What is an array variable? How does it differ from an ordinary variable?
An array variable or a subscripted variable is written along with array name followed by one or more subscript in brackets. An array variable is declared in type declaration statement along with the size of the array.
An ordinary variable or scalar variable is written without any subscript, and is always referring to a single value.

10. How do you initialize arrays in C ?
Arrays are initialized while they are declared in type declaration statement. Consider the following example:
```
int x[10] = {5, 25, 32, -30, 22};
```
This statement initializes `x[0] = 5, x[1] = 25, x[2] = 32, x[3] = -30` and `x[4] = 22`.
Elements `x[5]` to `x[9]` are not assigned any value.
```
int a[3][3] = {{1,0,0}, {0,1,0}, { 0,0,1 }};
```
This statement initializes a unit matrix of order 3 × 3. Note that each row of the matrix is enclosed in `{}`.