Contents

Unit 8
Arrays

8.0. Structure of the Unit

8.1. Unit Outcomes

After the successful completion of this unit, the student will be able to:

1. Explain the array declaration and operations on single and multi-dimensional arrays.
2. Apply arrays in search, sort, and other applications.
3. Discuss declaration and I/O operations on strings.
4. Develop programs on matrices, string handling functions, search, and sort problems.

8.2. Introduction to Arrays

An array is a single variable that stores multiple values of the same data type. An array is declared using the name of the array and the number of elements in square brackets. An example of array declaration is shown as follows:

```
int roll_list[10];
```

In this declaration, roll_list is a variable of the type int and it can store a maximum of 10 elements.

```
float a[3] = {10.5, 22.8, 12.9};
```

An array is declared and initialized with static values in the above declaration.
The array is accessed using an index as `printf ("%d", a[2]);`

a[0]	a[1]	a[2]
10.5	22.8	12.9

Here the array stores elements from the 0th position. Thus, the above statement will print the second element. The array element can be changed as shown below.

```
a[1] = 100;
```

The array `a` is a single-dimensional array. There can be multi-dimensional arrays in C programming. The array declaration includes a maximum size of an array. This size is estimated based on the problem or the application. Arrays are homogenous as one array can hold a list of items of only one data type.

Advantages and Drawbacks of Arrays: Arrays provide easy access with the help of an index. Arrays are useful in critical operations such as search, sorting, print, etc. With the use of multi-dimensional arrays, several matrix operations can be managed. A library of built-in functions can be created using arrays for common sorting and searching operations. Arrays can be used as parameters to a function and pointers are used to return the arrays from the function. The drawback of an array is that the size of an array is fixed, and it cannot be changed. In many situations, the size of the array is overestimated or falls short in some I/O operations.

A C program with a single-dimensional array is given in Program 1.

```
  #include <stdio.h>

  int main() {
      int list[10],i,n,sum = 0;
      printf("Enter the size of the list");
      scanf("%d",&n);
      printf("\n Enter the %d elements",n);
      for(i=0;i<n;i++)
      {
      scanf("%d",&list[i]);
      sum = sum + list[i];
      }
      printf("\n Sum of elements of a list is:%d", sum);
      return 0;
}
Sample Input/Output
  Enter the size of the list 4
  Enter the 4 elements 10 20 3 5
  10 20 3 5
   Sum of elements of a list is:38
```

Program 1: Addition of Elements of an Array

8.3. Operations on Arrays

The most common operations on arrays are: create, insert, delete, modify, and print the elements of an array. A C program that declares a single-dimensional array with insert and delete operations is given in Program 2.

```
  #include <stdio.h>
  int main() {
      int list[10],i,n, key,flag = 0;
      printf("Enter the size of the list");
      scanf("%d",&n);
      printf("\n Enter the %d elements",n);
      for(i=0;i<n;i++)
```

```
      scanf("%d",&list[i]);
      printf("enter element to be searched:");
      scanf("%d", &key);
      for(i=0;i<n;i++)
      {
          if(key == list[i])
          {
              flag = 1;
              break;
          }
      }
      if(flag==1)
      printf("\n Search is Successful");
      else
      printf("\n Search is Not Successful");
      return 0;
}
```

Sample Input/Output:

```
Enter the size of the list 4
4
Enter the 4 elements 12 3 7 8
12 3 7 8
enter element to be searched:3
Search is Successful
```

Program 2: Linear Search

In Program 2, a key element is searched with every element of an array from 0 to n-1. If the element is found, the variable flag is set to 1. If the flag value is 1, the key element is found in the list, or it is not present. This method is known as linear search.

When an element from an array is deleted, the memory is not deallocated, but only the value is erased. A typical C program to add and delete elements in an array is given in Program 3.

```
 #include <stdio.h>
 int main() {
      int list[10],i,n, pos;
      printf("Enter the size of the list");
      scanf("%d",&n);
      printf("\n Enter the %d elements",n);
      for(i=0;i<n;i++)
      scanf("%d",&list[i]);
      printf("enter position of element to be deleted:");
      scanf("%d", &pos);
      if(pos>n-1)
      printf("\n Deletion is not possible");
      for(i=pos-1;i<n-1;i++)
```

```
      {
          list[i] = list[i+1];
      }
      printf("\n Array after Deletion is:");
      for(i=0;i<n-1;i++)
      {
          printf("%d\n", list[i]);
      }
      return 0;
}
```

Sample Input/Output:

```
 Enter the size of the list 4
 4
 Enter the 4 elements 1 2 3 4
  1 2 3 4
 Enter position of element to be deleted:1
 Array after Deletion is:2 3 4
```

Program 3: Deleting an Element in an Array

In Program 3, the position of an element is taken as an input, and in a for loop the element is replaced by the next element. In this program, the last element's position is empty.

```
 #include <stdio.h>
 int main() {
      int list[10],i,n,j,temp=0;
      printf("Enter the size of the list");
      scanf("%d",&n);
      printf("\n Enter the %d elements",n);
      for(i=0;i<n;i++)
      scanf("%d",&list[i]);
      for(i=0;i<n;i++)
      {
          for(j=i;j<n;j++)
          {
              if(list[i]>list[j])
              {
                  temp = list[i];
                  list[i] = list[j];
                  list[j] = temp;

              }
          }
       }
      printf("\n Array after Sorting is:");
      for(i=0;i<n;i++)
      {
```

```
          printf("%d\n", list[i]);
      }
      return 0;
}
```

Sample Input/Output:

```
  Enter the size of the list 6
  6
  Enter the 6 elements 1 12 3 8 10 4
   1 12 3 8 10 4
  Array after Sorting is:1
  3
  4
  8
  10
12
```

Program 4: Sorting Elements of a List in an Ascending Order

In Program 4, the elements are sorted in an ascending order. Two for loops are used and at the end of every iteration of index i, the largest element is stored in its position.

8.4. Two-dimensional Arrays

Two-dimensional arrays are used to store large amounts of data. 2-D arrays can be used as matrices with a fixed number of rows and columns. A two-dimensional array is declared as follows.

```
int a[10][10];
```

This statement declares a two-dimensional array of size 10. For doing any operations on matrices there must be two loops, 1) for processing row elements and 2) for column elements. Commonly for loop is preferred for matrix operations. Read, print, or several other operations on 2D arrays are given in a few programs given below. The transpose of a matrix is logic given in Program 5.

```
  #include <stdio.h>
  int main() {
    int a[10][10], transpose[10][10], r, c;
    printf("Enter rows and columns: ");
    scanf("%d %d", &r, &c);

    // asssigning elements to the matrix
    printf("\nEnter matrix elements:\n");
    for (int i = 0; i < r; ++i)
    for (int j = 0; j < c; ++j) {
        scanf("%d", &a[i][j]);
    }

    // printing the matrix a[][]
    printf("\nEntered matrix: \n");
    for (int i = 0; i < r; ++i)
    {
    for (int j = 0; j < c; ++j)
    {
      printf("%d  ", a[i][j]);
    }
```

```
       printf("\n");
    }

    // computing the transpose
    for (int i = 0; i < r; ++i)
    for (int j = 0; j < c; ++j) {
      transpose[j][i] = a[i][j];
    }

    // printing the transpose
    printf("\nTranspose of the matrix:\n");
    for (int i = 0; i < c; ++i)
    for (int j = 0; j < r; ++j) {
      printf("%d  ", transpose[i][j]);
      if (j == r - 1)
      printf("\n");
    }
    return 0;
}
```

Sample Input/Output:

```
Enter rows and columns: 2 2
Enter matrix elements:
1 2 3 4
Entered matrix:
1  2
3  4

Transpose of the matrix:
1  3
2  4
```

Program 5: Two-dimensional Array for Transpose of a Matrix

Similarly, a C program to multiply two matrices is given in Program 6.

```
#include<stdio.h>
int main() {
    int a[10][10], b[10][10], c[10][10], n, i, j, k;

    printf("Enter the value of N (N <= 10): ");
    scanf("%d", & n);
    printf("Enter the elements of Matrix-A: \n");

    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            scanf("%d", & a[i][j]);
        }
    }
    printf("Enter the elements of Matrix-B: \n");
    for (i = 0; i < n; i++) {
```

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

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

        printf("The product of the two matrices is: \n");
        for (i = 0; i < n; i++) {
            for (j = 0; j < n; j++) {
                printf("%d\t", c[i][j]);
            }
            printf("\n");
        }
        return 0;
}
```

Sample Input/Output:

```
 Enter the value of N (N <= 10): 3
 Enter the elements of Matrix-A:
 1 2 3
 4 5 6
 7 8 9
 Enter the elements of Matrix-B:
 1 1 1
 2 2 2
 3 3 3
 The product of the two matrices is:
 14    14      14
 32    32      32
  50    50     50
```

Program 6: Matrix Multiplication using 2-dimensional Array

For the addition and subtraction of two matrices, we need to check the number of rows and columns. If these two values are not the same then, any operation cannot be performed on matrices. This is shown in Program 7. The same condition applies to the multiplication of matrices.

```
 #include <stdio.h>
  #define MAXROW      10
 #define MAXCOL      10
```

```
 /*User Define Function to Read Matrix*/
void readMatrix(int m[][MAXCOL],int row,int col)
{
    int i,j;
    for(i=0;i< row;i++)
    {
        for(j=0;j< col;j++)
        {
            printf("Enter element [%d,%d] : ",i+1,j+1);
            scanf("%d",&m[i][j]);
        }
    }
}

/*User Define Function to Read Matrix*/
void printMatrix(int m[][MAXCOL],int row,int col)
{
    int i,j;
    for(i=0;i< row;i++)
    {
        for(j=0;j< col;j++)
        {
            printf("%d\t",m[i][j]);
        }
        printf("\n");
    }
}

int main()
{
    int a[MAXROW][MAXCOL],b[MAXROW][MAXCOL],result[MAXROW][MAXCOL];
    int i,j,r1,c1,r2,c2;


    printf("Enter number of Rows of matrix a: ");
    scanf("%d",&r1);
    printf("Enter number of Cols of matrix a: ");
    scanf("%d",&c1);
    printf("\nEnter elements of matrix a: \n");
    readMatrix(a,r1,c1);
    printf("Enter number of Rows of matrix b: ");
    scanf("%d",&r2);
    printf("Enter number of Cols of matrix b: ");
    scanf("%d",&c2);
    printf("\nEnter elements of matrix b: \n");
    readMatrix(b,r2,c2);
    /*sum and sub of Matrices*/
    if(r1==r2 && c1==c2)
    {
        /*Adding two matrices a and b, and result storing in matrix
      result*/
```

```
          for(i=0;i< r1;i++)
          {
              for(j=0;j< c1;j++)
              {
                  result[i][j]=a[i][j]+b[i][j];
              }
          }

          /*print matrix*/
          printf("\nMatrix after adding (result matrix):\n");
          printMatrix(result,r1,c1);

          /*Subtracting two matrices a and b, and result storing in
        matrix result*/
          for(i=0;i< r1;i++)
          {
              for(j=0;j< c1;j++)
              {
                  result[i][j]=a[i][j]-b[i][j];
              }
          }

          /*print matrix*/
          printf("\nMatrix after subtracting (result matrix):\n");
          printMatrix(result,r1,c1);
       }
      else
      {
          printf("\nMatrix can not be added, Number of Rows & Cols
        are Different");
      }
      return 0;
}
Sample Input/Output
  Enter number of Rows of matrix a: 2
  Enter number of Cols of matrix a: 2

  Enter elements of matrix a:
  Enter element [1,1] : 1
  Enter element [1,2] : 2
  Enter element [2,1] : 3
  Enter element [2,2] : 4
  Enter number of Rows of matrix b: 2
  Enter number of Cols of matrix b: 2

  Enter elements of matrix b:
  Enter element [1,1] : 5
  Enter element [1,2] : 6
  Enter element [2,1] : 3
  Enter element [2,2] : 4
```

```
Matrix after adding (result matrix):
6       8
6       8

Matrix after subtracting (result matrix):
-4      -4
0       0
```

Program 7: Multiplication of Two Matrices

8.5 Introduction to Strings

In C programming string is used to handle non-numeric data. A String is a sequence of characters terminated with a null character '\0'. The sequence of characters is stored as an array of characters. The string is terminated by a unique null character unlike an integer or any other data type array. The declaration of the string variable is given below.

char string_name[size];
Example: `char student_name[20];`

A string can be initialized with constant data by various methods as given below.

- A string can be initialized without giving the size of the array as:
  `char student_name[] = "vaishali";`
- A string can be initialized with size, but one extra space must be reserved for the null character as:
  `char student_name[8] = "vaishali";`
- A string can be assigned by taking one character at a time as:
  `char student_name[8] = {'v','a','i','s','h','a','l','i','\0'};`
- A string can be initialized without giving size, here the compiler inserts the null character automatically.
  `char student_name[] = {'v','a','i','s','h','a','l','i','\0'};`

The above-mentioned data is stored in a string as given in Figure 8.1.

Index	0	1	2	3	4	5	6	7	8
Value	'V'	'a'	'i'	's'	'h'	'a'	'l'	'i'	'\0'
Memory Address	1000	1001	1002	1003	1004	1005	1006	1007	1008

Figure 8.1 String Data Type

A simple program for handling strings is shown in Program 8.

```
// C program to read a string from user
#include<stdio.h>

int main()
{
 // declaring string
 char str[50];
```

```
   // reading string
   scanf("%s",str);
   // print string
   printf("%s",str);
   return 0;
}
```
Sample Input/Output
```
Happy
Happy
```
Program 8: Reading and Printing of Strings

The above-shown program can display only one word which is without white spaces. The same program can be modified to read a sentence or line of characters as given in Program 9.

```
  #include<stdio.h>
  int main()
  {
      char str[100];
      printf("Enter any sentence: ");
      gets(str);
      printf("\nYou have entered:\n");
      puts(str);
      return 0;
}
Sample Input/Output
Enter any sentence: Happy New Year
You have entered:
Happy New Year
```
Program 9: gets() and puts() Function for Reading/Printing String

gets() and puts() functions are used for reading and printing strings with white spaces. The input is read till the ENTER key is pressed.

8.6. String-handling Functions

String manipulation is done using library functions defined in a header file string.h. The most used string functions are to:

- Find the length of a string
- Combine or concatenate two strings
- Copy one string into another
- Compare two strings
- Find the occurrence of a character in strings

These functions are summarized in Table 8.1.

Table 8.1. String Manipulation Functions

#	Name of the Function	Description
1	int strlen (string name)	It returns the number of characters in a string.
2	strcpy (Destination string, Source String);	It is for copying the source string into the destination string, provided the length of the destination string >= source string.

3	strncpy (Destination string, Source String, n);	It copies the 'n' characters of the source string into the destination string. The length of the destination string must >= that of the source string.
4	strcat (Destination String, Source string);	It combines two strings. The length of the destination string must be > than the source string.
5	strncat (Destination String, Source string,n);	This is used for combining or concatenating n characters of one string into another. The length of the destination string must be greater than the source string. The resultant concatenated string will be in the destination string.
6	int strcmp (string1, string2);	This function compares 2 strings. It returns the ASCII difference of the first two non–matching characters in both strings.
7	strncmp (string1, string2,2);	This function is used for comparing the first 'n' characters of 2 strings.

The functions strlen, strcpy, and strncpy are demonstrated in Program 10.

```
#include <string.h>
#include <stdio.h>
int main(void){
   char a[30] = "Hello";
   char b[30] = "Everybody";
   char c[30] = "";
   int l;
   l = strlen (a);
   printf ("length of the string = %d\n", l);
   strcpy(a,b);
   printf("The copied string in a is: %s\n",a);
   strncpy(c,b,5);
   printf("The copied string in a is: %s",c);
   return 0;
}
Sample Input/Output:
length of the string = 5
The copied string in a is: Everybody
The copied string in a is: Every
```

Program 10: strlen and strcat Functions

The functions strcat(), and strncat() are illustrated in Program 11.

```
#include <string.h>
#include <stdio.h>
int main(void){
   char a[30] = "Hello";
```

```
    char b[30] = "Everybody ";
    char c[30] = "Go home";
    strcat(a,b);
    printf("The combined string in a is: %s\n",a);
    strncat(b,c,2);
    printf("The copied string in a is: %s",b);
    return 0;
}
```
Sample Input/Output:
```
The combined string in a is: HelloEverybody
The copied string in a is: Everybody Go
```
Program 11: String Concatenation

The strcmp() and strncmp() functions are demonstrated in Program 12.

```
 #include <string.h>
 #include <stdio.h>
 int main(void){
    char a[30] = "Madam";
    char b[30] = "Madam";
    char c[30] = "Goodmorning";
    if(strcmp(a,b)==0)
    printf("Words a and b are same\n");
    else
    printf("Words a and b are not same");
    if(strcmp(a,c)==0)
    printf("Words a and c are same\n");
    else
    printf("Words a and c are not same");
    return 0;
}
```
Sample Input/Output:
```
Words a and b are the same
Words a and c are not the same
```
Program 12: strcmp() and strncmp() functions

8.7. Self-Assessment Questions/Activities

Q1. Discuss the features of arrays with an example. [5 Marks, L1]

Q2. Design and develop a C program to find the addition of elements of a matrix using an array. [10 Marks, L2]

Q3. Write a C program to search for a number and its position in an array. [10 Marks, L2]

Q4. Write a C program to reverse a string and check whether it is a palindrome. [8 Marks, L2]

Q5. Explain the advantages and drawbacks of an array. [5 marks, L2]

8.8. Multiple-Choice Questions

Q1. Array elements are stored in_____. [1 Mark, L2]

A. Random
B. Hierarchical
C. Sequential
D. Any order

Q2. Maximum number of index in C Program is____ [1 Marks, L1]

A. 2
B. 4
C. 10
D. There is no limit

Q3. Predict the output of the following_____. [1 Mark, L1]

```
void main()
{
  int a[25] = {1,2,3,4,5};
 printf("%d",a[5]);
}
```

A. 5
B. 0
C. 6
D. 10

Q4. The output of the following is_____ [1 Mark, L2]

```
void main()
{
  char s1[] = "abcd";
  char s2[] = "abcd";
if(s1==s2)
printf("equal");
else
printf("not equal");
}
```

A. Not Equal
B. Equal
C. Error
D. Garbage Value

Q5. The array is an example of _____ type memory allocation.

A. Compiler-time
B. Run-time
C. Random
D. Hierarchical

8.8. Keys to Multiple-Choice Questions

Q1. Sequential (C)

Q2. There is no limit (D)

Q3. 0 (B)

Q4. Not Equal (A)

Q5. Compile Time (A)

8.9. Summary of the Unit

An array is a collection of elements of the same data type. Each element is stored at a fixed position during compile time. Arrays are stored sequentially. The element of an array can be accessed using the index. Several operations such as search, insert, create, print, add, sort, etc. can be performed using arrays. Arrays are typically used for several string manipulations. A string is a sequence of characters terminated by a null character. String-related functions are defined in string.h header file.

8.10. Keywords:

- Single dimensional array
- Search and sorting
- Two-dimensional array
- Matrices
- Strings
- strlen, strcat, strcmp

8.11. Recommended Learning Resources

[1] Herbert Schildt. (2017). *C Programming: The Complete Reference*. 4th ed. USA: McGraw-Hill.