CHAPTER NINE: C- Programming: Arrays

Chapter Objectives

By the end of this chapter the learner should be able to

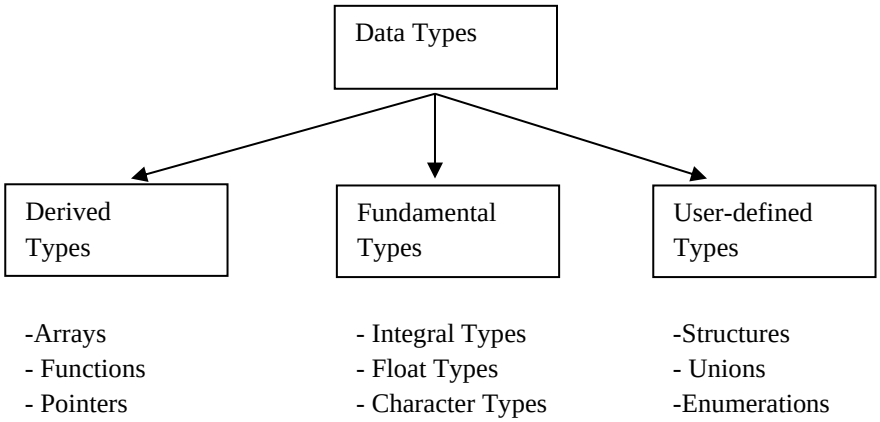
- Describe the C Program Data structures
- Declare and Initialize the C program Numeric Arrays
  - ✓ One-Dimensional
  - ✓ Two-Dimensional
  - ✓ Three Dimensional
- Declare and Initialize the C Program String Arrays
- Use String Arrays to manipulate Strings in C Program

9.1 Introduction

The data types so far encountered in C language, char, int, float, double are constrained by the fact that the variables of these types can store only one value at any given time. Thus they can only be used to handle limited amounts of data. C language supports a derived data type called **Array** that can be used in applications that need to handle large amounts of data.

An array is a fixed-size sequenced collection of elements of the same data type ie, a grouping of like-type data that share a common name. It can be used to represent a list of number or names. Since an array provides a convenient structure for representing data it is classified as one of the ***data structures*** in C.

9.2. Data structures in C



Arrays and structures are referred to as ***structured data types*** because they can be used to represent data values that have a structure of some sort. Structured data types provide an organizational scheme that shows the relationships among he individual elements and facilitate efficient data manipulations. These structured data types are called ***data structures.*** Other data structures in C includes;

- Stacks
- Linked-lists
- Queues
- Trees

Since an array can represent a list of items, the individual values of the items are referred to as elements.
Example
Salary[10] represent the tenth element (10^{th} employee salary in a list of organization employees salaries).

The ability to use a single name to represent a collection of items and to refer to an item by specifying the item number enables us to develop concise and efficient programs. We can use the arrays to represent not only a simple list of values but also tables of data in two, three or more dimensions, this giving rise to;

- One-dimensional arrays
- Two-dimensional arrays
- Three-dimensional arrays

9.3. One-Dimensional Arrays

A list of items can be given one variable name using only one subscript and such a variable is called a single-subscripted variable or a one-dimensional array. Example
x[1]. x[2], x[3] ……. x[n]. x[0] is allowed

To represent 35, 40, 20, 57, 19) by a array variable number, the array variable number is declared as

int number[5];

the computer will reserve five storage spaces as;

Number[0]	35
Number[1]	40
Number[2]	20
Number[3]	57
Number[5]	19

These elements may be used in programs just like any other C variable. Example

a = number[0] + 10;

number[4] = number[2] + number[3];

9.4. Declaration of One-Dimensional Arrays.

Arrays must be declared like other variable before they are used so that the computer can allocate space for them in memory. The declaration takes the following form;

type ***variable-name[size]***

Type specifies the type of elements that will be contained in the array such as int, float, char etc. Size indicates the maximum number of elements that can be stored inside the array.

float height[50]; **d**eclares an array height to contain 50 real (floating point) numbers

Note: the size should either be a numeric or a symbolic constant.

C language treats character strings simply as arrays of characters. The size in the character string represents the maximum number of characters that the string can hold. Example the string "WELL DONE" is stored in a character 'name array' as follows;

Char name[10];

'W'
'E'
'L'
'L'
' '
'D'
'O'
'N'
'E'
'\0'

A compiler always terminates a character string with a ***null character '\0'***. Thus when declaring character arrays, we must allow one extra element space for the ***null terminator***.

9.5. Initialization of One-Dimensional Arrays.

After an array has been declared, its elements must be initialized, otherwise they will contain garbage. Arrays can be initialized at the following stages: at compile time and at run time

9.5.1. Compile time initialization.

Format: **type *array-name[size] = { list of value };***

The values in the list are separated by commas

Example int number[3] = {0,0,0};

- Float total[5] = {0.0, 15.75, -10} will initialize the first three elements to 0.0, 15.75, -10.0 and the remaining two elements to zero

If the size is omitted example

int counter[] = { 1, 1,1,1}; the compiler allocates enough space for all initialized elements.

int number[3] = {20, 10, 30, 40}; is not allowed in C

9.5.2. Run Time Initialization

This approach is usually applied for initializing large arrays.

```
----------
----------
for ( i =1; i < 100; i = i+1)
{
  if  i < 50
    sum[i] = 0.0;
  else
   sum[i] = 1.0;
}
----------
--------
```

Program Example 9.1. Program for frequency counting

```
#include<stdio.h>
#include<conio.h>
#define  MAXVAL  50
```

```
#define COUNTER 11

main()
{
float  value[MAXVAL];
int  i, low, high;
int group[COUNTER] = {0,0,0,0,0,0,0,0,0,0,0};
printf("Enter a set of numbers Maximum 50 then press Enter\n");
                for (i = 0; i < MAXVAL; i++)
{
      scanf("%f", &value[i]);
  ++ group[ (int ) (value[i]) / 10];
}
printf("\n");
printf(" GROUP    RANGE      FREQUENCY\n\n");
for ( i = 0; i < COUNTER; i++)
{
        low = i *10;
                if (i == 10)
                   high = 100;
                else
                  high = low + 9;
 printf(" %2d    %3d to %3d      %d\n", i+1,   low,   high,   group[i]);
}
getch();
}
```

Note int group[COUNTER] = {0,0,0,0,0,0,0,0,0,0,0}; can be replaced with int group[COUNTER] = {0};

9.6. Searching and Sorting using Arrays

Searching and sorting are the two most frequent operations performed on arrays. Computer scientists have devised several data structures and searching and sorting techniques that facilitate rapid access to data stored in lists. Sorting is the process of arranging elements in the list according to their values, in ascending or descending order. A sorted list is called an ordered list. Sorted lists are especially important in list searching

because they facilitate rapid search operations. Many sorting techniques are available. The three simple and most important among them are;

- Bubble sort
- Selection sort
- Insertion sort.

Other sorting techniques includes Shell sort, Merge sort and Quick sort.

Searching is the process of finding the location of the specified elements in a list. The specified element is often called ***search key***. If the process of searching find a match of the search key with a list element value, the search is said to be successful otherwise it is unsuccessful. The two most commonly used search techniques are:

- Sequential search
- Binary search

Program Example 9.2. Program to sort Elements in an Array in Ascending order

```
#include<stdio.h>
#include<conio.h>
main()
{
  int arr[10],temp,i,j,n;
  printf("\nEnter any Value less Than 10 ");
   scanf("%d",&n);
  printf("\n\tEnter The Values into ARRAY ");
        for(i=0;i<n;i++)
        {
              printf("\n\n Enter Element no %d: ",i+1);
              scanf("%d",&arr[i]);
        }
              for(i=0;i<n-1;i++)
              {
              for(j=0;j<n-i;j++)
              {
                    if(arr[j] >arr[j+1])
                    {
```

```
                    temp=arr[j];
                    arr[j]=arr[j+1];
                    arr[j+1]=temp;
                    }
                }
                }
printf("\n\n-- Sorted Series --");
        for(i=0;i<n;i++)
        {
        printf("\n \n \t %d",arr[i]);
        }
        getch();
}
```

Test data 32, 43, 23, 57, 47, 34

9.7. Two – Dimensional Arrays

Two dimensional arrays can be used to store a table of values example

	Item1	Item2	Item3
Sales girl 1	310	275	365
Sales Girl 2	210	190	325
Sales Girl 3	405	235	240
Sales Girl 4	260	300	380

The table can be thought of as a Matrix containing 4 rows and 3columns. In mathematics we represents a matrix value by using two subscripts such as V_{ij} V denotes the entire matrix and V_{ij} refers to the value in the row (i) and column (j). example V_{31} = 405

Two dimensional arrays are declared as follows;

- **Type array_name[row_ size][column_size];**

Each dimension of the array is indexed from zero to its maximum size minus 1; the first index selects the row and the second index selects the column within that row.

Program example 9.3. Program to show; Total sales by each girl; Total value of each item sold and Grand total of sales of all items by all girls

```
#include<stdio.h>
#include<conio.h>
#define MAXGIRLS 4
#define MAXITEMS 3
main()
{
      int value[MAXGIRLS][MAXITEMS];
      int girl_total[MAXGIRLS], item_total[MAXITEMS];
      int i, j, grand_total;
      printf ("Input data\n");
      printf("Enter values, one at a time row-wise\n\n");
        for (i = 0 ;  i < MAXGIRLS ; i++)
            {
             girl_total[i] = 0;
               for (j = 0 ; j < MAXITEMS ; j++)
                  {
                     scanf("%d", &value[i][j]);
                       girl_total[i] = girl_total[i] + value[i][j];
                   }
             }
             /* Computing the Items totals */
                       for (j = 0 ; j < MAXITEMS ; j++)
                       {
                         item_total[j] = 0;
                         for (i =0 ; i < MAXGIRLS ; i++)
                          item_total[j] = item_total[j] + value[i][j];
                       }
               /* computing the Grand totals */
                     grand_total = 0;
                       for (i = 0 ; i < MAXGIRLS ; i++)
                       grand_total = grand_total + girl_total[i];
               /* printing results */
                printf("\n GIRLS  TOTALS \n\n");
```

```
                for (i = 0 ; i < MAXGIRLS ; i++);
                 printf("Salesgirl[%d] = %d\n", i+1, girl_total[i]);
                 printf("\n ITEM TOTALs\n\n");
                      for (j = 0 ; j < MAXITEMS ; j++)
                       printf("Item[%d] = %d\n", j+1, item_total[j]);
                   printf("\nGrand Total = %d\n", grand_total);
          getch();
          }
```

Program Example 9.4. /*Multiplication table using two-dimensional array: */

```
#include<stdio.h>
#include<conio.h>
#define ROW 5
#define COLOUMN 5
main()
{
        int row,coloumn,product[ROW][COLOUMN],i,j;
        printf("Multiplication table:\n\n");
        printf(" ");
        for(j=1;j<=COLOUMN;j++)
        {
                printf("%4d",j);
        }
        printf("\n");
        printf("----------------------\n");
        for(i=0;i<ROW; i++)
        {
                row=i+1;
                printf("%2d|",row);
                for(j=1;j<=COLOUMN;j++)
        {
        coloumn=j;
        product[i][j]=row*coloumn;
        printf("%4d",product[i][j]);
```

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

9.7. Initializing Two-Dimension Arrays

Like the one-dimension arrays, two-dimension arrays may be initialized by following their declaration with a list of initial values enclosed in braces. Example

int table[2][3] = { 0,0,0,1,1,1);

initializes the elements of the first row with zeros and the second row elements with one.

Or

Int table[2][3] = {{0,0,0}, {1,1,1}};

Or **Int table[2][3] = {**

{0,0,0},

{1,1,1}

}

9.8. Character Arrays and Strings

A string is a sequence of characters that is treated as a single data item.. Character strings are often used to build meaningful and readable programs. The common operations performed on character string includes;

- Reading and writing strings;
- Combining strings together
- Copying one string to another
- Comparing strings for equality.
- Extracting a portion of a string.

9.9. Declaring and Initializing String Valuables.

C does not support strings as a data type. C allows representing strings a character arrays. Format

char string_name [*size]*;

The size determines the number of characters in the string_name. example

char city[10] or char name[30]

The C compiler automatically supplies a null character (‘\0’) at the end of the string when it assigns a character string to a character array. Thus the array size should be equal to the required name size plus 1.

Initializing a string array;

char city[9] = "NEW YORK"; OR char city[9] = {'N', 'E', 'W', ' ', 'Y', 'O', 'R', 'K', '\O'};

C also permits initialization of a character string array without specifying the number of elements. In such a case, the size of the array is determined automatically based on the number of elements initialized. Example

char string[] = {'G', 'O', 'O', 'D', '\0'}; defines the array string as a five element string.

One can also declare a size much larger than the string size during initialization stage , example

char[10] = "GOOD";

The compiler initializes an array of 10 elements and places the word "GOOD" into it terminating with null character.

G	O	O	D	\0	\0	\0	\0	\0	\0

Illegal formats;

Declaration such as;

char string2 [3] = "GOOD".

char s1[4] = "abc";

char s2[4];

s2 = s1; will give an error as an array name cannot be used as the left operand of an assignment operator.

9.10. Terminating with Null Character.

- A string is not a data type in C but is considered a data structure stored in an array.
- The string is a variable-length structure and is stored in a fixed-length array.
- The array size is not always the size of the string and most often it is much larger than the string stored in it. Thus the last element of the array need not be represent the end of the string.
- The Null character serves as the "end-of-string" marker.

9.11. Reading Strings from Terminals

Using Scanf() function.

Format

char address[10];

scanf("%s", address);

Scanf function terminates its input on the first white space (blanks, tabs, carriage returns, form feed and new line) it finds.

Example entering the name "NEW YORK" will only result with the "NEW" being read into the array, since the blank space between the NEW and YORK words will cause the scanf to terminate.

The & symbol is not required before the variable name.

To read the word "NEW YORK" then we need to use two character arrays example

```
Char adr1[5], adr2[5];
scanf ("%s  %s", adr1, adr2);
```

" NEW" will assigned to adr1 while "YORK" will be assigned to adr2

One can also specify the field width in the scanf() function,

```
scanf("%ws", variable_name);
```

9.12. Reading a Line of Test

C supports a format specification known as the edit set conversion code %[ ], that can be used to read a line containing a variety of characters, including white spaces

```
char  line[80;
scanf("%[^\n]", line);
printf("%s", line);
```

will read a line of text from the keyboard and print it out as entered.

9.12.1. Using *getchar* and *gets* Function instead of Scanf() function

The getchar and gets functions methods used in C to read a string of text containing whitespaces. Gets function is available from <stdio.h> header file

The getchar function takes the following form;

```
char ch;
ch = getchar( );
```

The gets function takes the following form;

gets(str); where ***str*** is a string variable declared properly. Example

```
char line[80];
gets(line);
```

printf(“%s”, line);

Program Example 9.5 A program to read a line of text from the Keyboard.

```
#include<stdio.h>
#include<conio.h>
main()
{
char line[81], character;
int c;
c = 0;
        printf("Enter text. Press <Return> at end\n");
        do
        {
                character = getchar();
                line[c] = character;
                c++;
        }
        while (character != '\n');
        c = c-1;
        line[c] = '\0';
        printf("\n%s\n", line);
        getch();
}
```

9.13. Writing Strings to Screen.

The function in C use to write string to the screen is the ***printf*** function. The printf function takes the following form

printf(“%s”, variable_name);

Program Example 9.6: Writing strings using %s format

```
/* documentation  section*/
#include<string.h>
#include<stdio.h>
#include<conio.h>
main()
```

```
{
char country[15] = “United Kingdom”;
printf(“\n\n”);
printf(“-------------------------\n”);
printf(“%15s\n”, country);
printf(“%5s\n”, country);
printf(“%15.6\ns”, country);
printf(“%-15s\n”, country);
printf(“%15.0\ns”, country);
printf(“%.3s\n”, country);
printf(“%s\n”, country);
getch();
}
```

9.14. String Handling Functions

The string functions are in the <string.h> header file

Function	Action
Strcat()	Concatenates two string
Strcmp()	Compares two strings
strcpy	Copies one string over another
Strlen()	Finds the length of a string
Strupr()	Converts to upper case
Strlwr()	Converts to lower case

9.14.1. Strupr() function

Used to convert string into upper case.

Program Example 9.7. Program to Convert String to Upper case

```
/* documentation section*/
#include<string.h>
#include<stdio.h>
#include<conio.h>
main()
{
char mystring[10];
```

```
printf("Enter your string ");
scanf("%s", &mystring);
printf("string to upper case is %s", strupr(mystring));
getch();
}
```

9.14.2. Strcat() function

Function used to join two strings together. It takes the form

strcat(string1, string2);

String1 and string2 are character arrays. When the function ***strcat()*** is executed, string2 is appended to string1. It does so by removing null characters at the end of string1and placing string2 from there.

The size of the string1 should be large enough to accommodate the combined string (string1 + string2);

9.14.3. Strcmp() Function

The function compares two strings identified by the arguments and has a value of 0 if they are equal. If they are not, it has the numeric difference between the first non matching characters in the string. The function takes the following form;

strcmp(string1, string2);

String1 and string2 may be string variable or string constants example

strcmp(name1, name2); strcmp(name1, "John"); strcmp("Rom", "Ram");

strcmp("there", "their"); returns the value -9 which is the numeric difference between ASCII "i" and ASCII "r" (i – r) = -9.

9.14.4. Strcpy() Function

The function Copies / assigns the contents of string2 to string1. String2 may be a character array variable or a string constant. It takes the following g form

strcpy(string1, string2);

Example

Strcpy(city, "NAIROBI"); will assign the string "NAIROBI" to string variable city.

9.14.5 Strlen() Function

The function counts and returns the number of characters in a string. It takes the following form

n = strlen(string);

n = integer variable which receives the value of the length of the string.

Program Example 9.8: Program example using the string handling functions

```
/* documentation  section*/
#include<string.h>
#include<stdio.h>
#include<conio.h>
main()
{
 char s1[20], s2[20], s3[20];
  int x, k1, k2, k3;
printf("\n\n Enter two string constants \n");
printf("?");
scanf("%s  %s", s1, s2);

x = strcmp(s1, s2);                          /* compares s1 and  s2 */
            if (x !=0)
             {
                printf ("\n\n Strings not Equal  \n");
               strcat(s1, s2);                          /* joins s1 and  s2 strings */
            }
            else
             printf("\n\n Strings are equal \n");
            strcpy(s3, s1);                          /* copies s1 into s3 */

            k1 = strlen(s1);                          /* gets the length of s1 */
            k2 = strlen(s2);                          /* gets the length of s2 */
            k3 = strlen(s3);                          /* gets the length of s3 */
printf("\n s1 = %s\t length = %d characters \n", s1, k1);
printf("\n s2 = %s\t length = %d characters \n", s2, k2);
printf("\n s3 = %s\t length = %d characters \n", s3, k3);

getch();
}
```

Test data

London / London

Nairobi / City

Chapter Review Questions

1. Discuss how initial values can be assigned to a multidimensional array.
2. What is a data structure? and why is an array considered a data structure?
3. Write a program to read two matrices A and B and print the following
   a) A + B
   b) A – B
4. What is the error in the following program

```
main()
{
 int x;
float y[ ]
…….
}
```

5. Marks for a Programming Methodology Exam for a class of 50 students are as follows;

43	65	51	27	79	11	56	61	82	09	25	36	07	49	55
63	74	81	49	37	40	49	16	75	87	91	33	24	58	78
65	56	76	67	45	44	36	63	12	21	73	49	51	19	39
49	68	93	85	59										

a). Write a program to count the number of students belonging to each of the following groups of marks

0 -9, 10 – 19, 20 – 29, 30 – 39 …… 90-99, 100