PROGRAMMING IN C

MODULE 2

Summarize the definition, initialization and accessing of single and multi dimensional arrays, Develop programs using single and multidimensional arrays, Illustrate the concept of divide and conquer method in solving problems, Develop C programs to implement searching (linear search and binary search) and sorting (selection sort and quicksort) algorithms. Explain the representation of strings in C, Develop C programs to perform different operations on strings, Illustrate passing arrays as parameters to a function.

Introduction to Arrays

- An array is a data structure used to store and manage a collection of multiple data items under a single variable name.
- It solves the problem of declaring numerous individual variables for a series of data.
- An array can be visualized as a "list" of "items" or a "pill organizer" where elements are "joined together in a contiguous manner.
- Elements are accessed using a numeric index, which enables efficient use of loops for repetitive operations.

Definition: Arrays

- An array is a **linear** and **homogeneous** data structure.
- **Linear Data Structure:** Elements are stored in a single, contiguous block of memory. The entire array occupies a single, uninterrupted span of memory addresses, without any padding or separation between elements.
- **Homogeneous Data Structure:** All elements in an array must be of the same data type (e.g., int, float, char). The declared data type determines the memory size for each element.

Single Vs Multi-Dimensional Arrays

- **Single-Dimensional Array:** This is the simplest type, functioning as a linear list. It requires a single subscript to access its elements.
- **Multi-Dimensional Array:** These arrays have more than one set of square brackets in their declaration and are conceptualized as "arrays of arrays". A two-dimensional array, for example, is an array where each element is itself a one-dimensional array. This structure is ideal for representing data in a grid or tabular format, such as matrices or game boards.

Single-Dimensional Arrays

Declaration: The Blueprint of an Array

- **Syntax:** data_type array_name[array_size];
- data_type: The type of elements the array will store (e.g., int, float, char).
- array_name: The identifier for the array.
- array_size: A positive integer constant specifying the number of elements. The size is fixed at compile time.
- Declaration alone does not assign values; local arrays will contain "garbage values" until initialized

Single-Dimensional Arrays

Initialization: Assigning Values to the Blueprint

- Array initialization is the process of assigning values, which can be done at compile-time or run-time.

Compile-Time Initialization

- Values are assigned during declaration using curly braces {}.
- **Full Initialization:** All elements are explicitly assigned values. Example:

```
int numbers[9] = {10, 20, 30, 40, 50};
```

- **Partial Initialization:** If fewer values are provided than the declared size, the remaining elements are automatically set to 0. Example:

```
int values[9] = {1, 2};
```

Single-Dimensional Arrays

Initialization: Assigning Values to the Blueprint

- **Initialization Without Specifying Size:** The compiler automatically determines the size by counting the initializers. Example:

  int marks = {75, 80, 85};

- **String Initialization:** A character array can be initialized with a string literal. The compiler automatically adds a null terminator (\0) at the end, so the size is one greater than the number of characters in the string. Example:

  char city[5] = "Delhi";.

Single-Dimensional Arrays

Initialization: Assigning Values to the Blueprint

Run-Time Initialization

- Values are assigned during program execution, typically for large arrays or when values are unknown at compile time.
- This is commonly done using a loop with input functions like scanf().

Single-Dimensional Arrays

Accessing Elements: Navigating the Array

- Elements are accessed using the array name and a zero-based **index** enclosed in square brackets.
- The first element is at index 0, and the last element of an array of size N is at index N−1.
- **Physical Memory Layout:**

```
+-----+-----+-----+-----+-----+
| 100 | 200 | 300 | 400 | 500 |
+-----+-----+-----+-----+-----+
```

Index: 0 1 2 3 4

Address: 1000 1004 1008 1012 1016

sizeof(int) is 4 bytes and the starting address is 1000

Single-Dimensional Arrays

Programs

- Finding the Sum and Average of Array Elements

Enter 5 integers:

Enter element 1: 10

Enter element 2: 20

Enter element 3: 30

Enter element 4: 40

Enter element 5: 50

Sum of elements: 150

Average of elements: 30.00

Single-Dimensional Arrays

Programs

- **HomeWork**
- **Finding Maximum and Minimum Elements**
- Enter 5 integers:
- 45 72 29 90 61
- Maximum element: 90
- Minimum element: 29

Multi-Dimensional Arrays

Defining Multi-dimensional Arrays

- Multi-dimensional arrays are arrays that contain one or more arrays as elements.
- A two-dimensional (2D) array is an array of one-dimensional arrays, logically viewed as a grid of rows and columns.
- This structure is highly useful for storing tabular data like matrices.

Multi-Dimensional Arrays

Declaration and Initialization

- Multi-dimensional arrays are arrays that contain one or more arrays as elements.
- A two-dimensional (2D) array is an array of one-dimensional arrays, logically viewed as a grid of rows and columns.
- This structure is highly useful for storing tabular data like matrices.
- **Syntax for Declaration:**
  - type array_name[size1][size2]...[sizeN];

Multi-Dimensional Arrays

Declaration and Initialization

- **Initialization:** Each row is enclosed in a separate set of curly braces, improving readability. This is also necessary for partial initialization of specific rows.

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

- Each inner set of braces represents a logical row.
- Accessing Elements in Multi-dimensional Arrays
- Accessing an element requires a subscript for each dimension, e.g.,

```
array_name[row_index][column_index]
```

- Nested loops are the standard way to traverse all elements. The outer loop typically iterates through rows, and the inner loop iterates through columns.

Multi-Dimensional Arrays

Programs

- **Printing a 2D Array as a Matrix**
- The matrix is: 1 2 3 4 5

  10 20 30 40 50

  5 10 15 20 25

PROGRAMMING IN C

MODULE 2

Summarize the definition, initialization and accessing of single and multi dimensional arrays, Develop programs using single and multidimensional arrays, Illustrate the concept of divide and conquer method in solving problems, Develop C programs to implement searching (linear search and binary search) and sorting (selection sort and quicksort) algorithms. Explain the representation of strings in C, Develop C programs to perform different operations on strings, Illustrate passing arrays as parameters to a function.

Passing Arrays

- In C, you can **pass arrays as parameters to a function** in order to process them (like printing, summing, etc.).
- Function to calculate sum of array elements –Single Dimensional
- Print 2d Matrix Array – Multi- Dimensional

PROGRAMMING IN C

MODULE 2

Summarize the definition, initialization and accessing of single and multi dimensional arrays, Develop programs using single and multidimensional arrays, Illustrate the concept of divide and conquer method in solving problems, Develop C programs to implement searching (linear search and binary search) and sorting (selection sort and quicksort) algorithms. Explain the representation of strings in C, Develop C programs to perform different operations on strings, Illustrate passing arrays as parameters to a function.

String

- String is a sequence of characters stored in a contiguous block of memory and terminated by a null character(‘\0’).
- This null terminator is crucial as it signals the end of the string, allowing functions to know where the string ends.
- char str_array[] = "Hello";

String Operations

- String manipulation in C is done using functions from the <string.h> header file.
- Concatenation
  - strcat(str1, str2);
- Copying
  - strcpy(destination, source);
- Length Calculation
  - int length = strlen(str);
- Comparison
  - int result1 = strcmp(str1, str2);

Sample Programs