Structure and Union in C Language

Module 02 – Arrays and Structures in C Language

In C, both **arrays** and **structures** are used to store multiple pieces of data.

- An **array** is like a collection of variables, all of the same type (like integers, floats, etc.), stored in a continuous block of memory.

- A **structure** is a way to group different types of data (like integers, floats, strings, etc.) under one name. It's useful when you want to store related data that is of different types.

Key Differences Between Arrays and Structures:

1. **Arrays**: Store multiple elements of the **same type**.
   - Example: int arr[5] = {1, 2, 3, 4, 5};

2. **Structures**: Store multiple elements of **different types**.
   - Example: struct Person { char name[50]; int age; float height; };

Example: An Array of Structures

Suppose you want to store information for multiple people, like their name, age, and height. You can use an array of structures to do this:

```c
#include <stdio.h>
#include <string.h>

// Define the structure
struct Person {
    char name[50];
    int age;
    float height;
};

int main() {
    // Create an array of structures
    struct Person people[3];

    // Assign values to each person in the array
    strcpy(people[0].name, "John");
    people[0].age = 30;
    people[0].height = 5.9;

    strcpy(people[1].name, "Alice");
    people[1].age = 25;
    people[1].height = 5.5;

    strcpy(people[2].name, "Bob");
    people[2].age = 35;
    people[2].height = 6.0;

    // Print the information
    for (int i = 0; i < 3; i++) {
        printf("Person %d - Name: %s, Age: %d, Height: %.1f\n",
               i+1, people[i].name, people[i].age, people[i].height);
    }

    return 0;
}
```

Output:

```
Person 1 - Name: John, Age: 30, Height: 5.9
Person 2 - Name: Alice, Age: 25, Height: 5.5
Person 3 - Name: Bob, Age: 35, Height: 6.0
```

In this example:
- We defined a **structure** Person that holds name, age, and height.
- We created an **array of structures** people[3] to store information for three people.
- We accessed and printed the data for each person using the **dot operator (.)**.

Solved Example: Combination of Arrays and Structures in C

Problem: You are working on a student management system where you need to store information about multiple students. For each student, you want to store:
1. Name
2. Age
3. Marks in three subjects (Math, Science, and English)

You need to store this information for 5 students and display their data.

Solution:

```
#include <stdio.h>

// Define the structure for Student
struct Student {
    char name[50];
    int age;
    int marks[3];  // Array to store marks in three subjects
};

int main() {
    // Declare an array of 5 structures to store data of 5 students
    struct Student students[5];

    // Input data for 5 students
    for (int i = 0; i < 5; i++) {
        printf("Enter details for student %d\n", i + 1);
```

Introduction to 'C' Language

```
    // Input name, age, and marks for each student
    printf("Enter name: ");
    scanf("%s", students[i].name);  // Read name without spaces

    printf("Enter age: ");
    scanf("%d", &students[i].age);

    printf("Enter marks for Math, Science, and English: ");
    for (int j = 0; j < 3; j++) {
        scanf("%d", &students[i].marks[j]);
    }
}

// Display the data for all students
printf("\n--- Student Data ---\n");
for (int i = 0; i < 5; i++) {
    printf("\nStudent %d: \n", i + 1);
    printf("Name: %s\n", students[i].name);
    printf("Age: %d\n", students[i].age);
    printf("Marks: Math: %d, Science: %d, English: %d\n",
           students[i].marks[0], students[i].marks[1], students[i].marks[2]);
}

return 0;
}
```

Explanation:

1. **Structure Definition:**
   - The structure Student contains:
     - name[50]: to store the student's name (up to 49 characters).
     - age: to store the student's age.
     - marks[3]: to store marks for three subjects (Math, Science, English).
2. **Array of Structures:**
   - students[5]: an array of 5 structures to store details for 5 students.
3. **Input:**
   - **Name Input:** We use scanf("%s", students[i].name) to read the student's name. This will not handle spaces, meaning it can only read a single word for the name (e.g., "John" but not "John Doe").
   - **Age and Marks Input:** We use scanf() to read the age and marks for each student.
4. **Display:**
   - After reading the input, we use a loop to display the data for all students, showing each student's name, age, and marks.

Introduction to 'C' Language

Expected output

```
Enter details for student 1
Enter name: John
Enter age: 20
Enter marks for Math, Science, and English: 85 90 88

Enter details for student 2
Enter name: Alice
Enter age: 19
Enter marks for Math, Science, and English: 78 82 80

Enter details for student 3
Enter name: Bob
Enter age: 21
Enter marks for Math, Science, and English: 88 85 87

Enter details for student 4
Enter name: Carol
Enter age: 20
Enter marks for Math, Science, and English: 92 94 90

Enter details for student 5
Enter name: David
Enter age: 22
Enter marks for Math, Science, and English: 80 75 78

--- Student Data ---

Student 1:
Name: John
Age: 20
Marks: Math: 85, Science: 90, English: 88

Student 2:
Name: Alice
Age: 19
Marks: Math: 78, Science: 82, English: 80

Student 3:
Name: Bob
Age: 21
Marks: Math: 88, Science: 85, English: 87
```

Student 4:
Name: Carol
Age: 20
Marks: Math: 92, Science: 94, English: 90

Student 5:
Name: David
Age: 22
Marks: Math: 80, Science: 75, English: 78

Assignment

1. Student Grade Calculation:
Create a program to store and display student details such as name, roll number, and marks in 4 subjects (e.g., Math, Science, English, and History). Then calculate the total marks, percentage, and determine the grade (A, B, C, or D based on percentage).
Input:
• Name, Roll Number, Marks in 4 subjects.
• Calculate total marks and percentage.
Output:
• Display the student's name, roll number, total marks, percentage, and grade.

2. Library Book Management:
Create a program to manage a library system that stores information about books. Each book has a title, author, and number of pages. The program should allow adding books, displaying the list of books, and searching for a book by its title.
Input:
• Book Title, Author, Number of Pages.
• Add a book to the library collection.
• Display all books.
• Search for a book by title.
Output:
• Display the list of books or display the found book's details.