

Array of Structures vs. Array within a Structure in C/C++

Array within a Structure

A structure is a data type in C/C++ that allows a group of related variables to be treated as a single unit instead of separate entities. A structure may contain elements of different data types – int, char, float, double, etc. It may also contain an array as its member. Such an array is called an array within a structure. An array within a structure is a member of the structure and can be accessed just as we access other elements of the structure.

Below is the demonstration of a program that uses the concept of the array within a structure. The program displays the record of a student comprising the roll number, grade, and marks secured in various subjects. The marks in various subjects have been stored under an array called marks. The whole record is stored under a structure called a candidate.

```
// C++ program to demonstrate the  
// use of an array within a structure  
  
#include <iostream>  
using namespace std;  
  
// Declaration of the structure candidate  
struct candidate {  
    int roll_no;  
    char grade;  
  
    // Array within the structure  
    float marks[4];  
};  
  
// Function to displays the content of  
// the structure variables  
void display(struct candidate a1)  
{  
  
    cout << "Roll number : " << a1.roll_no << endl;  
    cout << "Grade : " << a1.grade << endl;  
    cout << "Marks secured:\n";  
    int i;  
    int len = sizeof(a1.marks) / sizeof(float);
```

```

// Accessing the contents of the
// array within the structure
for (i = 0; i < len; i++) {
    cout << "Subject " << i + 1 << " : " << a1.marks[i] << endl;
}
}

// Driver Code
int main()
{
    // Initialize a structure
    struct candidate A
        = { 1, 'A', { 98.5, 77, 89, 78.5 } };

    // Function to display structure
    display(A);
    return 0;
}

```

Output:

```

Roll number : 1
Grade : A
Marks secured:
Subject 1 : 98.50
Subject 2 : 77.00
Subject 3 : 89.00
Subject 4 : 78.50

```

Array of Structures

An array is a collection of data items of the same type. Each element of the array can be int, char, float, double, or even a structure. We have seen that a structure allows elements of different data types to be grouped together under a single name. This structure can then be thought of as a new data type in itself. So, an array can comprise elements of this new data type. An array of structures finds its applications in grouping the records together and provides for fast accessing. Below is the demonstration of an array of structures. The array holds the details of the students in a class. The details include the roll number, grade, and marks, which have been grouped under a structure (record). There exists one record for each student. This is how a collection of related variables can be assembled under a single entity for enhancing the clarity of code and increasing its efficiency.

```
// C program to demonstrate the  
// usage of an array of structures  
#include <stdio.h>
```

```
// Declaring a structure class  
struct class {  
    int roll_no;  
    char grade;  
    float marks;  
};
```

```
// Function to displays the contents  
// of the array of structures  
void display(struct class class_record[3])  
{  
    int i, len = 3;  
  
    // Display the contents of the array  
    // of structures here, each element  
    // of the array is a structure of class  
    for (i = 0; i < len; i++) {  
        printf("Roll number : %d\n",  
            class_record[i].roll_no);  
        printf("Grade : %c\n",  
            class_record[i].grade);  
        printf("Average marks : %.2f\n",  
            class_record[i].marks);  
        printf("\n");  
    }  
}
```

```
}
```

```
// Driver Code
```

```
int main()
```

```
{
```

```
    // Initialize of an array of structures
```

```
    struct class class_record[3]
```

```
        = { { 1, 'A', 89.5f },
```

```
            { 2, 'C', 67.5f },
```

```
            { 3, 'B', 70.5f } };
```

```
    // Function Call to display
```

```
    // the class_record
```

```
    display(class_record);
```

```
    return 0;
```

```
}
```

Output:

```
Roll number : 1
```

```
Grade : A
```

```
Average marks : 89.50
```

```
Roll number : 2
```

```
Grade : C
```

```
Average marks : 67.50
```

```
Roll number : 3
```

```
Grade : B
```

```
Average marks : 70.50
```