Array of Structures in C

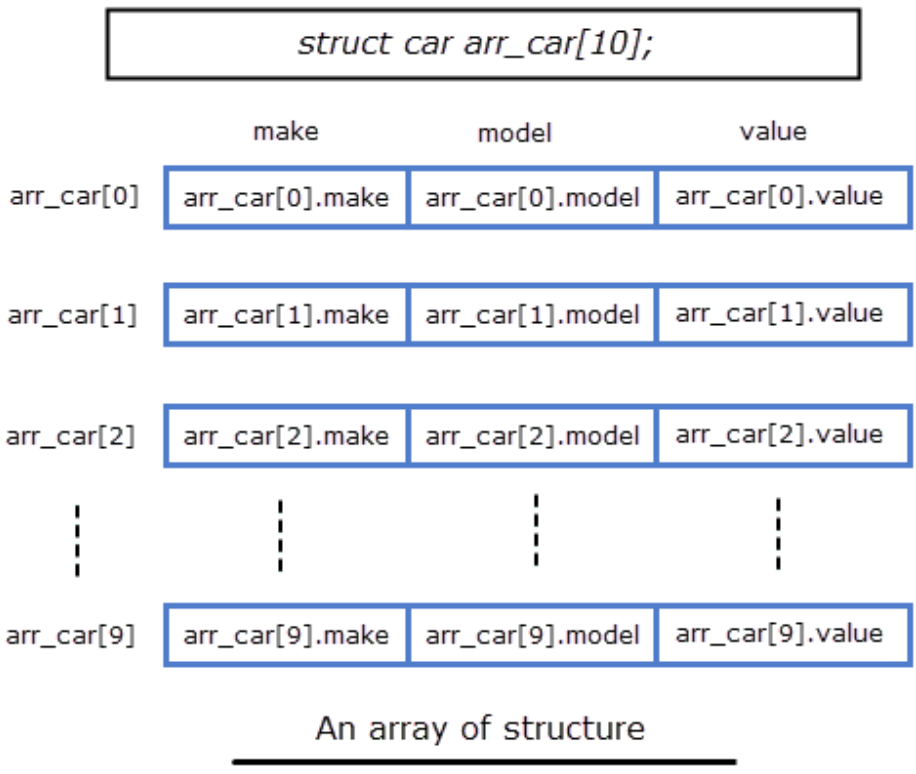
Declaring an array of structure is same as declaring an array of fundamental types. Since an array is a collection of elements of the same type. In an array of structures, each element of an array is of the structure type.

Let's take an example:

```
struct car
{
    char make[20];
    char model[30];
    int year;
};
```

Here is how we can declare an array of `structure car`.

```
struct car arr_car[10];
```



An array of structure

Here `arr_car` is an array of `10` elements where each element is of type `struct car`. We can use `arr_car` to store `10` structure variables of type `struct car`. To access individual elements we will use subscript notation (`[]`) and to access the members of each element we will use dot (`.`) operator as usual.

```
arr_stu[0] : points to the 0th element of the array.
arr_stu[1] : points to the 1st element of the array.
```

and so on. Similarly,

```
arr_stu[0].name : refers to the name member of the 0th element of the
array.
arr_stu[0].roll_no : refers to the roll_no member of the 0th element of the
array.
arr_stu[0].marks : refers to the marks member of the 0th element of the
array.
```

Recall that the precedence of `[]` array subscript and dot(`.`) operator is same and they evaluates from left to right. Therefore in the above expression first array subscript(`[]`) is applied followed by dot (`.`) operator. The array subscript (`[]`) and dot(`.`) operator is same and they evaluates from left to right. Therefore in the above expression first `[]` array subscript is applied followed by dot (`.`) operator.

Example:

```
#include<stdio.h>
#include<string.h>
#define MAX 2

struct student
{
    char name[20];
    int roll_no, i;
    float marks;
};

int main()
{
    struct student arr_student[MAX];
    int i;

    for(i = 0; i < MAX; i++ )
    {
        printf("\nEnter details of student %d\n\n", i+1);

        printf("Enter name: ");
        scanf("%s", arr_student[i].name);

        printf("Enter roll no: ");
        scanf("%d", &arr_student[i].roll_no);

        printf("Enter marks: ");
        scanf("%f", &arr_student[i].marks);
    }

    printf("\n");

    printf("Name\tRoll no\tMarks\n");

    for(i = 0; i < MAX; i++ )
    {
        printf("%s\t%d\t%.2f\n",
        arr_student[i].name, arr_student[i].roll_no, arr_student[i].marks);
    }

    // signal to operating system program ran fine
    return 0;
```

```
}
```

Expected Output:

```
Enter details of student 1

Enter name: Jim
Enter roll no: 1
Enter marks: 44

Enter details of student 2

Enter name: Sim
Enter roll no: 2
Enter marks: 76

Name Roll no Marks
Jim  1       44.00
Sim  2       76.00
```

How it works ?

In lines 5-10, we have declared a structure called the `student`.

In line 14, we have declared an array of structures of type `struct student` whose size is controlled by symbolic constant `MAX`. If you want to increase/decrease the size of the array just change the value of the symbolic constant and our program will adapt to the new size.

In line 17-29, the first for loop is used to enter the details of the student.

In line 36-40, the second for loop prints all the details of the student in tabular form.

Initializing Array of Structures

We can also initialize the array of structures using the same syntax as that for initializing arrays. Let's take an example:

```
struct car
{
    char make[20];
    char model[30];
    int year;
};
struct car arr_car[2] = {
                            {"Audi", "TT", 2016},
                            {"Bentley", "Azure", 2002}
                        };
```