Programming in C

Dr. Rupali P. Patil

Department of Electronics and Telecommunications

Module 4:
Structures And Unions

Structures And Unions

- Introduction, Declaring and defining Structure,
- Structure Initialization,
- Accessing and Displaying Structure Members,
- Array of Structures,
- Introduction to Unions,
- Structure Vs Unions

Arrays

An array is a sequence of data item of homogeneous values (same type).

Arrays are of two types:

1. One dimensional arrays (1D array)
2. Multidimensional arrays (2D, #D, etc.)

1D array

	Mon 0	Tue 1	Wed 2	Thu 3	Fri 4
Route 0	8	12	9	7	10
Route 1	5	7	3	0	4
Route 2	20	15	18	21	14
Route 3	6	5	5	8	11

2D array

Structures And Unions

- Structure is a group of variables of different data types represented by a single name.

- Syntax of struct

```
struct structureName
{
        dataType member1;
        dataType member2;
        ...
};
```

Structures And Unions

- Variable declaration of a structure

```
struct structureName
{
        dataType member1;
        dataType member2;
        ...
};
int main()
{
struct structureName var_name;
return 0;
}
```

OR

```
struct struct_name
{
   DataType member1_name;
   DataType member2_name;
   DataType member3_name;
   …
} var_name;
```

Structures And Unions

- Accessing members of a structure

var_name.member1_name;
var_name.member2_name;
…

Structures And Unions

- Accessing members of a structure

var_name.member1_name;
var_name.member2_name;
…

- Assign values to structure members

1) Using Dot(.) operator

var_name.memeber_name = value;

2) All members assigned in one statement

struct struct_name var_name =
{value for memeber1, value for memeber2
…so on for all the members}

Structures And Unions

Task

- WAP to display student name, roll number and CGPI using structures.
- WAP to display employee name, ID and year of experience.

- **Write separate code to Assign value at compile time and runtime**

Structures And Unions

- Array of Structures

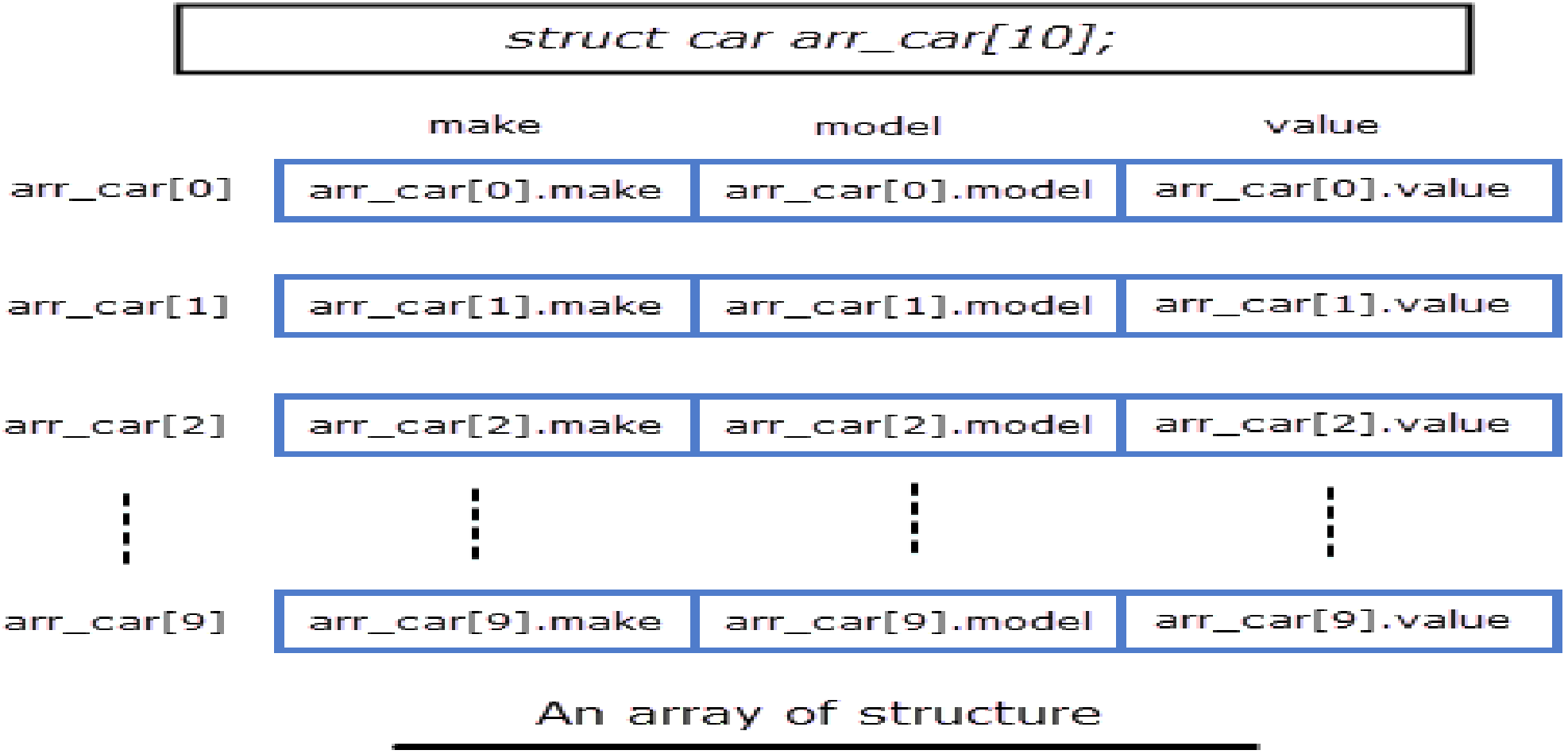
Declaring an array of structure is same as declaring an array of fundamental types. Since an array is a collection of elements of the same type. In an array of structures, each element of an array is of the structure type.

Syntax

```
struct car
{
    char make[20];
    char model[30];
    int year;
};
struct car arr_car[10];
```

Structures And Unions

- Array of Structures



Structures And Unions

- Array of Structures

Task

- WAP to display car name and number for 5 cars using array of structures

- Define array size
- Write for loop to accept as well as to display elements.

Structures And Unions

- Array of Structures

Task

- WAP to display car name and number for 5 cars using array of structures

- Define array size
- Write for loop to accept as well as to display elements.

```
for(i = 0; i < 5; i++ )
{
   printf("%s\t%d\n", arr_car[i].name, arr_car[i].number);
}
```

Structures And Unions

- Unions
- A union is a special data type available in C that allows to store different data types in the same memory location.

- You can define a union with many members, but only one member can contain a value at any given time.

- Unions provide an efficient way of using the same memory location for multiple-purpose.

Structures And Unions

- Unions

```
union [union tag]
{
   member definition;
   member definition;
   ...
   member definition;
} [one or more union variables];
```

Structures And Unions

- Unions

```
union [union tag]
{
   member definition;
   member definition;
   ...
   member definition;
} [one or more union variables];
```

```
union Data
{
   int i;
   float f;
   char str[20];
} data;
```

Structures And Unions

- Unions

```
union Data
{ int i;
  float f;
  char str[20];
} data;
```

Now, a variable of type Data can store an integer, a floating-point number, or a string of characters. It means a single variable, i.e., same memory location, can be used to store multiple types of data. You can use any built-in or user defined data types inside a union based on your requirement.

The memory occupied by a union will be large enough to hold the largest member of the union. For example, in the above example, Data type will occupy 20 bytes of memory space because this is the maximum space which can be occupied by a character string.

Structures And Unions

- Unions

```
#include <stdio.h>
#include <string.h>
union Data {
   int i;
   float f;
   char str[20];
};
int main( )
{
   union Data data;

   data.i = 10;
   data.f = 220.5;
   strcpy( data.str, "C Programming");

   printf( "data.i : %d\n", data.i);
   printf( "data.f : %f\n", data.f);
   printf( "data.str : %s\n", data.str);
   return 0;
}
```

```
#include <stdio.h>
#include <string.h>
union Data {
   int i;
   float f;
   char str[20];
};
int main( )
{
   union Data data;

   data.i = 10;
   data.f = 220.5;
   strcpy( data.str, "C Programming");

   printf( "data.i : %d\n", data.i);
   printf( "data.f : %f\n", data.f);
   printf( "data.str : %s\n", data.str);
   return 0;
}
```

Here, we can see that the values of i and f members of union got corrupted because the final value assigned to the variable has occupied the memory location and this is the reason that the value of str member is getting printed very well.

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

union Data {
   int i;
   float f;
   char str[20];
};
int main( ) {

   union Data data;
   data.i = 10;
   printf( "data.i : %d\n", data.i);

   data.f = 220.5;
   printf( "data.f : %f\n", data.f);

   strcpy( data.str, "C Programming");
   printf( "data.str : %s\n", data.str);
   return 0;
}
```

Here, all the members are getting printed very well because one member is being used at a time.

Structures And Unions • Structures Vs Unions

	STRUCTURE	UNION
Keyword	The keyword **struct** is used to define a structure	The keyword **union** is used to define a union.
Size	When a variable is associated with a structure, the compiler allocates the memory for each member. The size of structure is **greater than or equal to the sum of sizes of its members.**	when a variable is associated with a union, the compiler allocates the memory by considering the size of the largest memory. So, size of **union is equal to the size of largest member.**
Memory	Each member within a structure is assigned unique storage area of location.	Memory allocated is shared by individual members of union.
Value Altering	Altering the value of a member will not affed other members of the structure.	Altering the value of any of the member will alter other member values.
Accessing members	Individual member can be accessed at a time.	Only one member can be accessed at a time.
Initialization of Members	Several members of a structure can initialize at once.	Only the first member of a union can be initialized.

Arrays, Structures And Unions

4	**Arrays, Structures And Unions**		
	4.1	**Arrays:** Introduction to One Dimensional Arrays, Multidimensional Arrays, Declaration and Initialization of Arrays, Reading and Displaying arrays	**04**
	4.2	**Character Arrays and Strings:** Introduction, Declaring and Initializing String Variables, Reading Character and Writing Character, Reading and Writing Strings, String Handling Functions	**04**
	4.3	**Structures and Unions:** Introduction, Declaring and defining Structure, Structure Initialization, Accessing and Displaying Structure Members, Array of Structures, Introduction to Unions, Structure Vs Unions	**04**