

Shaping "skills" for "scaling" higher...!!!

WELCOME, PROGRAMMERS



01.

What is Structure?



WHAT IS STRUCTURE?





STRUCTURE



A structure is a **user-defined data type** that allows you to **group together variables of different data types** under a single name.

It provides a way to **represent** a **record** or a **collection of related data items**.

The structure allows you to define a composite data type that contains members with different data types.



Syntax of how to create a Structure



```
struct structureName
{
   dataType variable1;
   dataType variable2;
   // many more ...
};
```



Example of creating a structure

```
0 0 0
```

```
struct Car
   int carId;
   char carName[50];
   char carModel[50];
   char carColor[50];
```



Accessing members of a Structure



Using Simple Object

struct Car c1;

// c1.name

Using Pointer Object

struct Car *c1;

// c1->name



Accessing members of a Structure



```
void main()
    struct Car c1; // create a variable/object of a structure
   c1.id = 1;
    strcpy(c1.name, "Tata");
   strcpy(c1.model, "Harrier");
   strcpy(c1.color, "Black");
```



Print members of a Structure



```
void main()
   printf("%d\n", c1.id);
   printf("%s\n", c1.name); // Tata
   printf("%s\n", c1.model); // Harrier
   printf("%s\n", c1.color); // Black
```



We can create a variable/object of a structure by using two different approaches...



Creating a variable/object of a structure o o

```
struct Car
   int carId;
   char carName[50];
   char carModel[50];
   char carColor[50];
}c1, c2;
```

OR

```
void main()
   struct Car c1, c2;
```



Let's see the **Array of Objects** in structure...



Creating an Array of Objects

```
0 0 0
```

```
struct Car
   int carId;
   char carName[50];
   char carModel[50];
   char carColor[50];
```

```
void main()
     struct Car cars[100];
     // cars[0], cars[1], cars[2], ...,
     cars[99]
```



Let's see how many memory space acquired by a structure...



Knowing a size acquired by Structure

154



```
struct Car
{
   int carId;
   char carName[50];
   char carModel[50];
   char carColor[50];
};
```

RAM °				
	carId	carName	carModel	carColor
Memory Space	4 bytes	50 bytes	50 bytes	50 bytes

```
struct Car c1;
printf("%d", sizeof(c1));
Output:
```



02.

What is Union?



WHAT IS UNION?





UNION



A union is a user-defined data type that allows you to store different data types in the same memory location.

Unlike structures, where each member has its own memory space, **members of a union share the same memory location**.

This means that a union variable can hold values of only one member at a time.



Syntax of how to create an Union



```
union unionName
   dataType variable1;
   dataType variable2;
   // many more ...
```











































Example of creating an union

```
0 0 0
```

```
union Car
   int carId;
   char carName[50];
   char carModel[50];
   char carColor[50];
```



Accessing members of an Union

```
0 0 0
```

```
void main()
   union Car c1; // create a variable/object of an union
   c1.id = 1;
   strcpy(c1.name, "Tata");
   strcpy(c1.model, "Harrier");
   strcpy(c1.color, "Black");
```



Print members of an Union



```
void main()
   printf("%d\n", c1.id);
   printf("%s\n", c1.name);
   printf("%s\n", c1.model);
   printf("%s\n", c1.color);
```

Note: When you assign a value to one member of the union, the values of the other members become indeterminate.



















Difference between Structure & Union oo

Parameters	Structure	Union
Keyword	A user can deploy the keyword struct to define a Structure.	A user can deploy the keyword union to define a Union.
Internal Implementation	The implementation of Structure in C occurs internally- because it contains separate memory locations allotted to every input member.	In the case of a Union, the memory allocation occurs for only one member with the largest size among all the input variables. It shares the same location among all these members/objects.
Accessing Members	A user can access individual members at a given time.	A user can access only one member at a given time.





Difference between Structure & Union oo

Parameters	Structure	Union
	The Syntax of declaring a Structure in C is:	The Syntax of declaring a Union in C is:
Syntax	struct [structure name]	union [union name]
	{	{
	type element_1;	type element_1;
	type element_2;	type element_2;
	} variable_1, variable_2,;	} variable_1, variable_2,;
Size	A Structure does not have a shared location for all of its members. It makes the size of a Structure to be equal to the sum of the size of its data members.	A Union does not have a separate location for every member in it. It makes its size equal to the size of the largest member among all the data members.





Difference between Structure & Union oo







Parameters	Structure	Union	
Value Altering	Altering the values of a single member does not affect the other members of a Structure.	When you alter the values of a single member, it may affects the values of other members.	
Storage of Value	In the case of a Structure, there is a specific memory location for every input data member. Thus, it can store multiple values of the various members.	In the case of a Union, there is an allocation of only one shared memory for all the input data members. Thus, it stores one value at a time for all of its members.	
Initialization	In the case of a Structure, a user can initialize multiple members at the same time.	In the case of a Union, a user can only initiate the first member at a time.	



03.

What is Enumeration?





ENUMERATION?





ENUMERATION



Enumeration (enum) is a **user-defined data type** that consists of a **set of named integral constants**, known as **enumerators**.

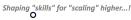
It provides a way to **associate names with numbers**, making the code more readable and understandable.



Syntax of how to create an Enumeration $\circ \circ \bullet$

```
enum enumName
{
    enumerator1,
    enumerator2,
    // additional enumerators
};
```





Example of creating an enumeration



Example of creating an enumeration

```
enum Days
   Sunday,
   Monday = 5, // 5
   Tuesday,
   Wednesday
```

Note: The numbering starts from 0 by default, but you can explicitly assign values to enumerators if needed.



























Variable/Object of an Enumeration

```
\circ \circ \bullet
```

```
void main()
{
    enum Days today;

    today = Wednesday; // Assigning the value Wednesday (7) to
    the variable today
}
```





Let's start now...





0