Structure

- A structure can be defined as a single entity holding variables of different data types that are logically related to each other.
- Structure is a user-defined data type in C language which allows us to combine data of different types together.
- Structure is a collection of variables of similar or different types under a single name.
- A structure is a collection of related data items (not necessarily of the same type) in which each identified by its own identifier, each of which is known as a member of the structure."

Defining Structure:-

- Structure defines a new data type which is a collection of primary (int, float, char etc.,) and derived data types (arrays and pointers).
- The struct keyword is used to define the structure.
- **The syntax of defining a structure is:-**

```
  struct <structure_name>
{
  data-type member1;
  data-type member2;
  ………………………………
  ……………………………..
};
```

Description of the Syntax:-

- **Keyword struct:** The keyword struct is used at the beginning while defining a structure.
- **structure_name:** This is the name of the structure which is specified after the keyword struct.
- **data-type:** The data type indicates the type of the data members of the structure. A structure can have data members of different data types.
- **member:** This is the name of the data member of the structure. Any number of data members can be defined inside a structure. Each data member is allocated a separate space in the memory.

Example 1:-

```
struct emp
{
   char ename[20];
    int eno;
   float esal;
};
```

Example 2:-

```
struct address
{
   char name [30];
   char street [20];
   char city [15];
   char state [15];
   int pincode;
};
```

Declaring Structure Variables:-

- The structure definition does not actually create variables. Instead, it defines data type only.
- When a structure is defined, no storage or memory is allocated.
- To allocate memory of a given structure type and work with it, we need to create variables.
- Structure variable declaration is similar to the declaration of any normal variable of any other data type.
- Structure variables can be declared in many two ways:

1. Declaration of Structure Variables with Structure Definition:-

This way of declaring a structure variable is suitable when there are few variables to declared.

Syntax:-

```
   struct <structure_name>
{
   data-type member1;
```

```
    data-type member2;
    …………………
   ……………………
} struct var1, struct_var2;
```

Example:-

```
  struct emp
{
  char ename[20];
  int eno;
  float esal;
} e1, e2;
```

2. Declaration of Structure Variables Separately:-

This way of creating structure variables is preferred when multiple variables are required be declared. The structure variables are declared outside the structure.

Syntax:-

```
  struct <structure_name>
{
  data-type member1;
  data-type member2;
};
    struct <structure_name> struct_var1, struct_var2, ....;
```

Example:-

```
  struct emp
{
  char ename[20];
  int eno;
  float esal;
};
   struct emp e1, e2, e3;
```

The structure variables can be declared inside a main() function as shown below:

```
void main()
{
  struct emp e1, e2, e3;
}
```

Initializing Members of Structure:-

- Structure members cannot be initialized like other variables inside the structure definition.
- This is because when a structure is defined, no memory is allocated to the structure's data members.
- Memory is allocated only when a structure variable is declared. Let us consider the below code.

Example:-

```
  struct emp
{
  char ename[20]="Naveen" ;          // COMPILER ERROR
  int eno=1001;                      // COMPILER ERROR
  float esal=16500.00;               // COMPILER ERROR
} e1, e2;
```

Example:-

```
#include<stdio.h>
#include<conio.h>
struct emp
{
  char ename[20];
   int eno,
   float esal;
}
   void main()
{
   struct emp e1;
  strcpy(e1.name, "Naveen");
   e1.eno=1001;
   e1. esal-16500.00;
   getch( );
}
```

Accessing Members of Structure:-

- The members of a structure are accessed outside the structure by the structure variables using the dot operator (.).
- The following syntax is used to access any member of a structure by its variable the general syntax is:
  <structure_variable>. <structure_member>

Example:-

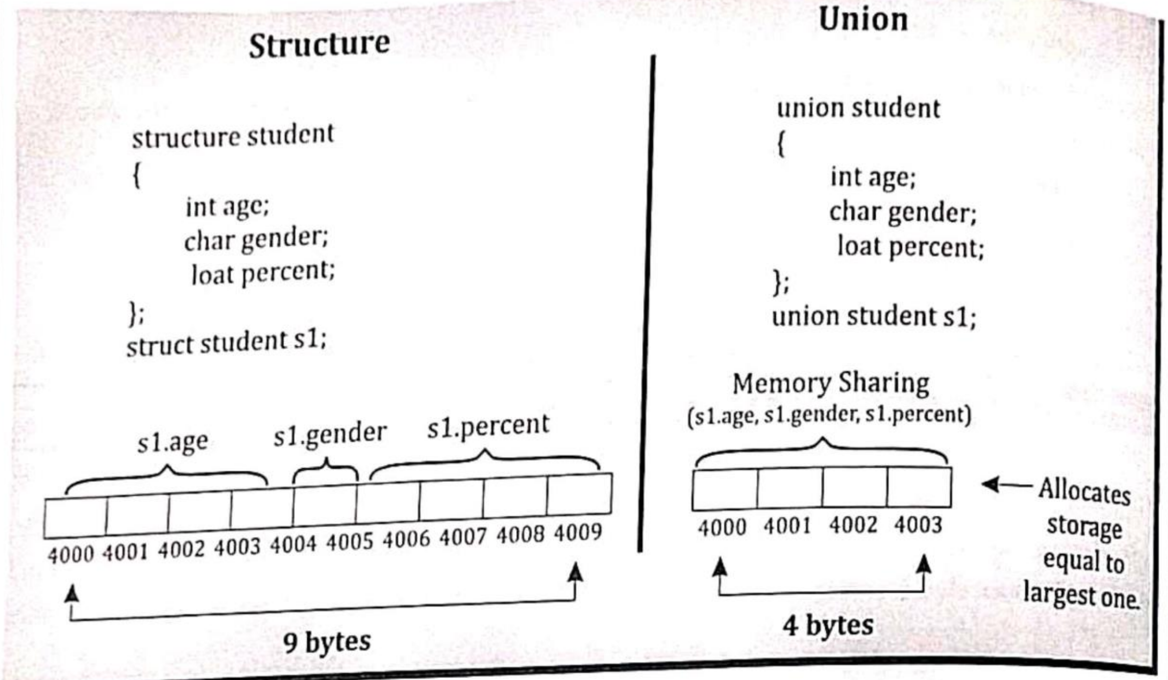
Program To Define, Assign and Access the members of structure:-

```
#include<stdio.h>
#include<conio.h>
void main( )
{
   struct emp
  {
     char ename[20];
     int eno;
     float easl;
  }
    struct emp e1;
    strcpy(e1.ename, “SRIKANTH”);
    e1.eno = 511;
   e1.easl = 20000.50;
   printf(“employee name of e1 is %s\n “,  e1.ename);
   printf(“employee number of e1 is %d\n “,  e1.eno);
   printf(“employee salary of e1 is %0.2f  “,  e1.esal);
   getch( );
}
```

Output:-

```
employee name of e1 is SRIKANTH
employee number of e1 is 511
employee salary of e1 is 20000.50
```

Conceptual Memory View of Structures and Unions



Structure

```
structure student
{
      int age;
      char gender;
       loat percent;
};
struct student s1;
```

Union

```
union student
{
      int age;
      char gender;
       loat percent;
};
union student s1;
```

11.13 Difference between Structure and Union

Structure	Union
1. The struct keyword is used to define a structure	1. The union keyword is used to define a a Union
2. Memory is allocated for each member of the structure. Every member has its own memory.	2. Memory is allocated as per largest member of the union. All members use the same memory.
3. Altering the value of a member will not affect other members of the structure	3. Altering the value of any of the member will alter the other member values.
4. The maximum memory size allocated is greater than or equal to the sum of the sizes memory of all the individual declared.	4. The maximum memory size allocated is equal to the size of the larger member.
5. All the individual members can be accessed at a time.	5. Only one member can be accessed at a time.
6. More storage space is required.	6. Minimum storage space is required.
7. It may be initialized with all its members	7. Only its first member may be initialized.