Structure

- Structure is a user-defined data type.
- Structure stores logically related (similar or non-similar) elements in contiguous memory location.
- Structure members can be accessed using "." operator via struct variable.
- Structure members can be accessed using "->" operator via struct pointer.
- Size of struct = Sum of sizes of struct members.
- If struct variable initialized partially at its point of declaration, remaining elements are initialized to zero.

```c
// struct data-type declaration (global or local)
struct emp {
    int empno;
    char ename[20];
    double sal;
};
// struct variable declaration
struct emp e1 = {11, "John", 20000.0};
// print struct members
printf("%d%s%lf", e1.empno, e1.ename, e1.sal);
```

- A variable of a struct can be member of another struct.
- This can be done with nested struct declaration.

```c
struct emp {
    int empno;
    char ename[20];
    double sal;
    struct {
        int day, month, year;
    }join;
};
```

```c
struct date {
    int day, month, year;
};
struct emp {
    int empno;
    char ename[20];
    double sal;
    struct date join;
};
struct emp e = { 11, "John", 2000.0, {1, 1, 2000} };
printf("%d %s %d-%d-%d\n", e.empno, e.ename, e.join.day, e.join.month,
e.join.year);
```

SUNBEAM

Structure padding

- For efficient access compiler may add hidden bytes into the struct called as "struct padding" or "slack bytes".
- On x86 architecture compiler add slack bytes to make struct size multiple of 4 bytes (word size).
- These slack bytes not meant to be accessed by the program.
- Programmer may choose to turn off this feature by using #pragma.
    - #pragma pack(1)

```c
struct test {
    int a;
    char b;
};
printf("%u\n", sizeof(struct test));

#pragma pack(1)
struct test {
    int a;
    char b;
};
printf("%u\n", sizeof(struct test));
```

Bit Fields

- A bit-field is a data structure that allows the programmer to allocate memory to structures and unions in bits in order to utilize computer memory in an efficient manner.
- Bit-fields can be signed or unsigned.
- Signed bit-field, MSB represent size + or -.
- Unsigned bit-field, all bits store data.
- Limitations of bit-fields
    - Cannot take address of bit-field (&)
    - Cannot create array of bit-fields.
    - Cannot store floating point values.

```c
struct student {
     char name[20];
     unsigned int age: 7;
     unsigned int roll: 6;
};
struct student s1 = { "Ram", 10, 21 };
printf("%s, %d, %d", s1.name, s1.age, s1.roll);
```

SUNBEAM