Structure, Union, and Enum in C++

1. Structure (`struct`)

- A **structure** is a user-defined data type that groups different data types together under one name.
- Each member of a structure has its own memory.
- **Syntax:**

```
struct Student {
    int rollNo;
    char name[50];
    float marks;
};
```

Key Points:

- Members are **stored in separate memory locations**.
- Total memory = sum of sizes of all members.
- Access members using `.` operator:

```
Student s1;
s1.rollNo = 101;
s1.marks = 88.5;
```

2. Union (`union`)

- A **union** is also a user-defined data type but with a key difference:
    - **All members share the same memory location.**
- Only **one member can hold a value at a time**.
- **Syntax:**

```
union Data {
    int intVal;
    float floatVal;
    char charVal;
};
```

Key Points:

- Memory allocated = size of the **largest member**.
- Changing one member's value affects the others.
- Access:

```
Data d;
d.intVal = 10;
d.floatVal = 12.5;  // overwrites intVal
```

3. Enumeration (`enum`)

- An **enum** is a user-defined type consisting of a set of named integral constants.

- Useful for making code more readable.
- Syntax:

```
enum Color { Red, Green, Blue };
```

Key Points:

- Default values start from `0, 1, 2…`
- You can assign custom values:

```
enum Weekday { Mon = 1, Tue, Wed, Thu, Fri, Sat, Sun };
```

- Access:

```
Color c = Green;   // c = 1
Weekday w = Fri;   // w = 5
```

Difference Between Structure and Union

Feature	Structure	Union
Memory	Each member has its own storage	All members share the same memory
Size	Sum of all members	Size of the largest member
Usage	Used when multiple values needed together	Used when only one value needed at a time

Summary:

- **Structure** → group different data, all active together.
- **Union** → save memory, only one member active at a time.
- **Enum** → define symbolic names for integral values.