C++ User-Defined Data Types

Exploring `struct`, `union`, and `enum` for custom data organization.

The `struct` Keyword

Structure (`struct`)

- A **user-defined data type** that groups related variables (members) of **different data types** under a single name.
- Keyword used for declaration: ***`struct`**.
- Each member of a structure is allocated its **own separate memory location**.
- It acts as a blueprint for a record (like a database row).

```
struct  {
     ;
     ;
    // ...
};
```

`struct` Example: Student Details

```
 #include
#include
using namespace std;

struct student {
    string name;
    int roll_no;
    float marks;
};

int main() {
    struct student s1;
    s1.roll_no = 1;
    s1.name = "Rahul";
    s1.marks = 89.5;

    cout << "Student Details:" << endl;
    cout << "Name: " << s1.name << endl;
    cout << "Roll No: " << s1.roll_no << endl;
    cout << "Marks: " << s1.marks << endl;

    return 0;
```

The `union` Keyword

Union (`union`)

- Similar to a `struct`, but **all members share the same memory location**.
- It is designed for memory efficiency when you only need to use **one member at a time**.
- The size of a `union` is equal to the size of its **largest member**.
- When you assign a value to one member, it **overwrites** the data of the other members.

```
union  {
     ;
     ;
    // ...
};
```

`union` Example: Data Overwriting

```
 #include
using namespace std;

union Data {
    int intvalue;
    float floatvalue;
    char charvalue;
};

int main() {
    union Data d;

    d.intvalue = 10; // 1. Assign int
    cout << "Integer: " << d.intvalue << endl;

    d.floatvalue = 12.5; // 2. Assign float (overwrites int)
    cout << "Float: " << d.floatvalue << endl;

    d.charvalue = 'A'; // 3. Assign char (overwrites float/int)
    cout << "Character: " << d.charvalue << endl;

    cout << "After assigning char, intvalue becomes: " << d.intvalue << endl;
```

The `enum` Keyword

Enumeration (`enum`)

- Used to assign names to **integer constants** (enumerators).
- The primary goal is to make the code easier to **read** and **maintain**.
- Keyword used for declaration: **`enum`**.
- By default, the first enumerator starts at **0**, and subsequent enumerators automatically **increment by 1**.

```
enum  {
    , // = 0
    , // = 1
    , // = 2
    // ...
};
```

`enum` Example: Weekday

```
 #include
using namespace std;

enum weekday {
    sunday,     // Value 0 (by default)
    monday,     // Value 1
    tuesday,    // Value 2
    wednesday,  // Value 3
    thursday,   // Value 4
    friday,     // Value 5
    saturday    // Value 6
};

int main() {
    weekday today = wednesday;

    cout << "Day number of wednesday: " << today << endl;
    // Output: Day number of wednesday: 3

    return 0;
}
```

Summary: `struct` vs. `union`

Feature	`struct` (Structure)	`union` (Union)
Memory Allocation	**Separate** memory for each member.	**Shared** memory for all members.
Size	Sum of the sizes of all members.	Size of the largest member.
Data Integrity	All members hold their values simultaneously.	Only one member's value is valid at any given time.
Purpose	To group related data of different types (a record).	To save memory when only one piece of data is needed at a time.

Questions?

Ready to move on to the next topic!