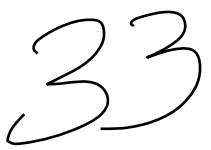
Let’s solve it



Structures and Unions

Que: Basic structure of C program

Does not mean "structures → User defined Data Type"

Keyword 'struct'

Documentation
Link section
Global storage class
Function declarations / prototypes
int main()
{
    // Declaration
    // Execution section
}
function definition.

→ Keywords 'struct' & 'union'

They help you/developer create user defined datatype.

Developer

C lang.

Datatypes

- Primary Data Types
- Derived Data Types
- User Defined Data Types
  'struct' & 'union'

Important for Data Type

- will have name
  i.e student, email,
  any entity defined by certain characteristics.

- will have memory

- operations

How to use a datatype?

Via creating variables.

---

i.e. Fruit itself does not exist.

```
struct Student
{
    int id;
    char name[10];
    int marks;
};
```

- `struct Student` ← name of Datatype
- `{ ... }` ← characteristics
- members of struct
- `};` ← definition of struct Student.

There is no memory allocated yet.

Creating variable of a structure 'student' datatype

datatype variablename;

struct student s1, s2, s3; Has now memory. Its very much like

user defined datatype

mango is a fruit.

Instance of a fruit

typedef struct student) stud ;

stud s1, s2, s3;

|| storage classes concept is applied to all structure variables.

one more way of creating structure variable

```
struct employee
{
  int id;
  char name[10];
  long long int salary;
} s1,s2,s3;
```

defining or creating structure

You can create variable/s also.

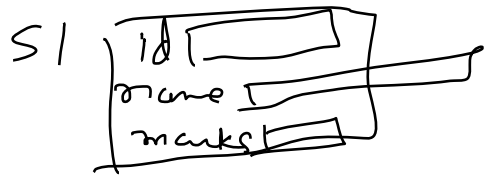
s1, s2, s3 → variables

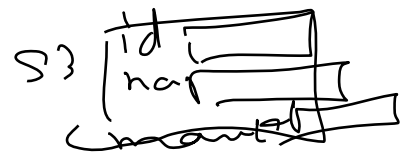
If above is written outside of all functions, the storage clss applied for s1, s2, s3 is global.

store and display 3 student's information

```
struct student
{
    ---;   ✓ data
    ---;   ✓ members
};

int main()
{
    struct student s1, s2, s3;   // local variables (automatic)
    printf("Enter student info");
    scanf("%d %s %d", &s1.id, s1.name, &s1.marks);
```





dot operator connects struct variable to its members

s2. s2. s2.

s3. s3. s3.

// Display

printf("%s %d %d\n", s1.name, s1.id, s1.marks);

printf("%s %d %d\n", s2.name, s2.id, s2.marks);

printf("%s %d %d", s3.name, s3.id, s3.marks);

}

Array of structures

struct student s1, s2, s3;

struct student s[3];

```
for(i=0; i<3; i++)
{
    scanf("%d %s %d", &s[i].id, s[i].name, &s[i].marks);
}

for(i=0; i<3; i++)
{
    printf("%s %d %d", s[i].name,
                       s[i].id
                       s[i].marks);
}
```