

Let's solve it

39

Rules for initializing

→ We can NOT initialize individual members inside the structure template (blueprint/plan of a building)

struct student

{
~~int id = 5;~~ // Not allowed; syntax error;
char name[10];

};

struct student s1, s2, s3;
 id id id
 □ □ □
 garbage

- The order (sequence) of values enclosed in braces must match the order of members in the structure definition/template.

```
struct student
{
  1 int id;
  2 char name[10];
  3 int marks;
};
```

int main()
{
 struct student s1 = { 5, "John", 96 };
 struct student s2 = { "John", 5, 96 };
}

~~name is not fine~~

Diagram illustrating the order of values in structure initialization:

- For `s1`, the values `5`, `"John"`, and `96` are assigned to `id`, `name`, and `marks` respectively, matching the order in the structure definition.
- For `s2`, the values `"John"`, `5`, and `96` are assigned to `name`, `id`, and `marks` respectively. This order does not match the member order in the structure definition, which is marked as incorrect.

- It is permitted to have a partial initialization.
We can initialize only the first few members and leave the remaining blank.

✓ student st = { 5 };

= { 5, "John" };

= { "John", 99 } ; X

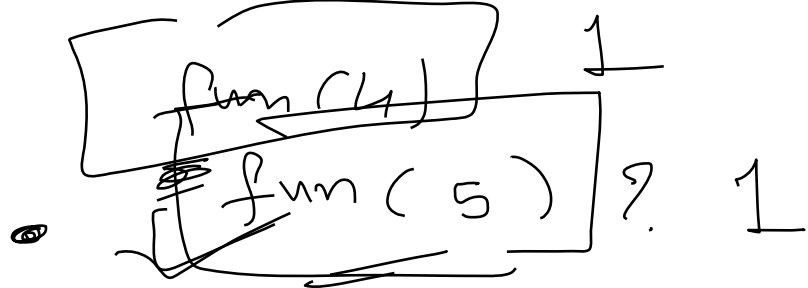
3.5. You may change the entries into structure template.

- The uninitialized members,
will be assigned default values
as below:

- Zero for integer & floating (numbers)
- NULL '\0' for characters and
strings.

Qwz
Recursion

```
int fun (int x)
{
    static int cc = 0;
    x++;
    x++; // 4
    if (x < 5) // 1 < 5
    {
        fun(x); // 4
        // fun(x); // b
        // fun(x); // c
    }
    return cc;
}
```



```
int main()
{
    int y = 0;
    int c = fun(y); // main call
    printf("%d", c);
    return 0;
}
```

Program counter register

microprocessor.

