

1

```
int id;  
char grade;  
float marks;
```

Змебѣ

g.)

```
struct student s1;  →  | | | |  ← id 4B
                        | | | |  ← grade 1B
                        | | | |  ← mark 4B
```

↑ one variable/object of student

```
st.id = 101;  
st.grade = 'A';  
st.marks = 85.9;
```

```
printf("C\n id=%d, grade=%c, marks=%f", s2.id, s2.grade, s2.marks);
```

② struct student s1 = {101, 'A', 85.9}; // Initialization during declaration or object creation.

③ Partial Initialization

struct student s1 = {101}; ✓ Id's value.
s1 = {101, 'A'};

```

④ st.id = 101;
  st.marks = 859;

  printf("In id = %d, grade = %c, marks = %f", st.id, st.grade, st.marks);

```

Note:- We must initialize all members/variable.

Array of structure:-

Example

```
struct student std[3]; // int a[3];
```

↓ ↓ ↓
Data type variable size 100

strict student $\underline{s_1, s_2, s_3}$; // $s_4, s_5, s_6, \dots, s_{100}$

Std[0] →

Id
grade
Marks

 101
A
95.9

std[i] →

rd	102
grade	B
marks	98.8

std[2] →

id
grade
marks

 103

st $\rightarrow$

id
grade
Marks

S2 →

id
grade
Marks

s3 →

id
grade
Marks

```
printf("id=%d, grade=%f, marks=%f", s1.id, s1.grade, s1.marks);
printf("id=%d, grade=%f, marks=%f", s2.id, s2.grade...
```

150

for (int i=0; i<=100; i++) How to print?

```

    printf("id = %d, grade = %c, marks = %f", std[i].id, std[i].grade, std[i].marks);
}

```

= x =

Initialize Array of structure directly

```

struct student
{
    int id;
    char grade;
    float marks;
};

```

```

struct student std[3] = {
    { 101, 'A', 85.5 },
    { 102, 'B', 79.9 },
    { 103, 'A', 89.3 }
};

```

0	id	}	4
1	id		
2	id		
3	id		
4	grade	}	1
5			
6			
7		}	8 Byte
8	marks		
9	marks		
10	marks	}	4
11	marks		

$4 + 1 + 3 + 4$
 $4 + 4 = 8$
 (12)

Memory Concept

```

struct student std[3];
std[0] → 12 Byte
std[1] → 12 Bytes
std[2] → 12 Bytes

```