Unions

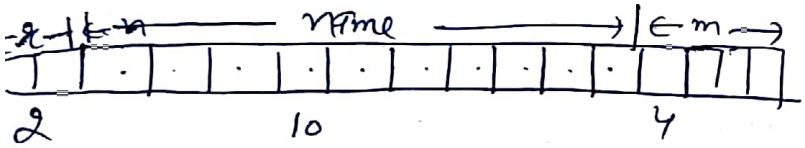
- Unions are also heterogeneous collection of elements like structure.
- It uses union keyword to define a union.

```
union student
{
  int sr;
  char name[10];
  float marks;
} S1;
```

- It uses in hardware designing mostly where memory saving is efficient.
- Unions are used to conserve memory.

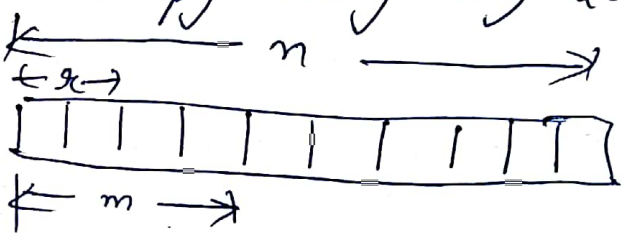
Structure	Unions
- heterogeneous collection of elements	- heterogeneous collection of elements.
- It uses struct keyword	- It uses union keyword
- It is mostly used in software development	- It is mostly used in hardware implementation.
- In structure all members occupy individual memory	- In unions, only one member at a time occupy the memory and memory allocation will be of largest data type.

Eg
```
struct student
{
int x;          → 2
char name[10];  + 10
float m;        + 4
};
```
16 bytes allocated




```
union student
{
int x;          2
char name[10];  10
float m;        4
};
```
largest data type = 10 (name)
So 10 bytes will be allocated

and each member will occupy memory one by one.



typedef (User-defined data type)

- It is used to create a user defined data type
- Basically, it is not a new data type but it merely gives a new name to some existing data type. Eg

typedef <old data type> <new data type>

```
typedef int age;
```

⟹ Now int variable has given a new name age. Now you can declare a variables with age data type instead of int.

```
age male, female;
```

- With this, you can give meaningful name to your data type

⟹ Read 'egs. from book.

Bitfields :-

If in a program, a variable is to take only two values — 1 and 0, we really need only a single bit to store it.

Ily, if a variable is to take values from 0 to 3, then two (2) bits are sufficient to store these.

2 values (0, 1) → 0 1

0 - 3 values (0,1,2,3) → 0 0
0 1
1 0
1 1
] 2 bits reqd to store

- Bit fields are members of structure.
- We cannot have array of bit fields.
- Bit fields are always unsigned.

how to define bit fields

unsigned <variable name> : No. of bits.

unsigned int a : 1 [1-bit] allocation

unsigned int b : 2 [2-bits]

```
struct employee
{
    unsigned int rollno : 4      → 4 bits
    unsigned int marks : 7       → 7 bits
    unsigned int b : 2           → 2 bits
};
```

otherwise, we have to take rollno as 2 bytes or 16 bits → But if we know max. value it will occupy of 4 bits (0-16) → then, we can use bit fields.

They are used to save memory.

→ Read eg. from book.