~~Structure~~

Structure

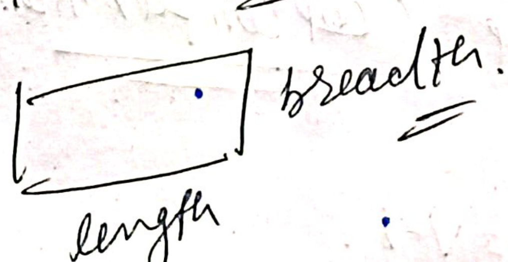
* It is defined as collection of data member or related data members under one name; and those ~~#~~ data member may be of similar type ~~of~~ or different types.

so usually it is defined as collection of similar or dissimilar data members under one name.

* Structure is used to define user defined data types.

Definining structure
Size of structure
Declaring of structure
Accessing members

example rectangle




```
struct Rectangle{
  int length;  -> 2 byte
  int breadth; -> 2 byte
};
```

so 4 byte memory it takes.

rectangle is define by group of value. length and breadth

by just defining structure it does not take any memory if we create a variable of this type then it will occupy that much memory.

how to declare a variable of type structure inside main function

```
int main()
{ struct Rectangle r;        // declaration
  struct Rectangle r = {10, 5};  → initialize with value.
```

this is declaration plus initialization

r.
l | 10/15
b | 5/10

accessing members

```
r.length = 15;    // dot operator is used of accessing members.
r.breadth = 10;

printf(" Area of rectangle is %d", r.length * r.breadth
```

example on structure. (complex numbers)

complex number

a + ib   i = $\sqrt{-1}$.
   ↑

~~Structure~~

Struct complex
{
int real; → 2
int img; → 2
};

Total = 4 byte

2: Student

Struct Student
{ int roll; — 2
char name [25]; — 25
char dept [10]; → 10
char address [50]; → 50
};

Total = ~~77~~ byte

accessing

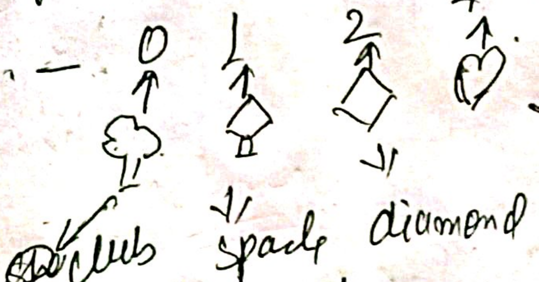
Struct Student · S;
S. roll = 10;
S. name = "john";
⋮

Example ~~Play~~ Playing card.

it has following values

1. face value = 1, 2, . . . 10, J, Q, K (A = 1, J = 11, Q = 12, K = 13) → lets say we give code for it.

Shape. — 0, 1, 2, 4

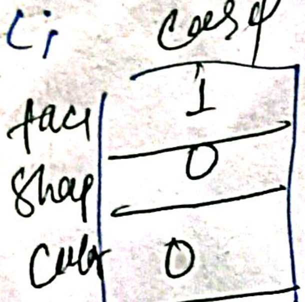


colour → 0 black, 1 red

Struct card
{ int face;
int shape;
int colour;
};

int main ()
{ struct card c;
c. face = 1;
c. shape = 0;
c. color = 0;

card



if we have initalize value of all 52 card then we can do this by Array.

each card value

Ex

```
int main()
{ struct card deck[52] = { {1,0,0}, {2,0,0}
  --- {1,1,0}, {2,1,0} };
  Printf("%d", deck[0].face);
  Printf("%d", deck[0].shape);
}
```