Structure

Arrays are used to represent a homogeneous collection of values. Similarly, a structure is another user defined data type available in Rust that allows us to combine data items of different types, including another structure. A structure defines data as a key-value pair.

Declaring structure

The struct keyword is used to declare a structure.

```
struct Employee {
    name: String,
    age: u8
}
```

Initializing structure

```
let employee_1 = Employee {
        name: String::from("Abhishek"),
        age: 17
    };
    println!("{}", employee_1.name);
    println!("{}", employee_1.age);
```

Modifying structure

```
   let mut employee_1 = Employee {
        name: String::from("Abhishek"),
        age: 17
    };
    employee_1.name = String::from("Kumar");
```

Passing a struct to a function

```
struct Employee {
    name: String,
    age: u8
}

fn input_struct(emp: Employee) {
    println!(" fun - {}", emp.name);
    println!(" fun - {}", emp.age);
}

fn main() {

    let mut employee_1 = Employee {
        name: String::from("Abhishek"),
        age: 17
    };
    println!("{}", employee_1.name);
    println!("{}", employee_1.age);

    input_struct(employee_1);
}
```

Methods in structure

Methods are like functions .Methods are declared with the fn keyword. The scope of a method is within the structure block. Methods are declared outside the structure block. The impl keyword is used to define a method within the context of a structure. The first parameter of a method will be always self.

```
struct Rectangle{
    length:i32,
    width:i32
}
// this is method in structure.
impl Rectangle{
    fn area(&self)->i32{
        (self.length)*(self.width)
    }
}

fn main(){
    let rec_1 = Rectrangle{
        length:10,
        width:20
    };
    println!("{}",rec_1.area());

}
```