



```
/
/Cargo.toml          # [ ] [ ] [ ] [ ] [ ]

/src                  # [ ] [ ] [ ] [ ]
└-- /main.rs         # main
└--

/target              # [ ] [ ] [ ] [ ] [ ]
└-- /debug           # debug [ ] [ ]
└-- /release         # release [ ] [ ]
```

```
[package]
name = "project_name"
version = "0.1.0"
edition = "2023"

[dependencies]
```



```
# [ ] [ ] [ ] [ ] [ ] [ ]
let x = 5;

# [ ] [ ] [ ] [ ] [ ]
let mut x = 5;
x=10;

# [ ] [ ] [ ] [ ]
```

```

let x = 9;

fn function_name(){
}

// 00000
// 00 ->00000
fn fun_name() -> u32 {
    // 00000000
    // 55
    // 0000 return
    // return 55;
}

```

0000

```

struct Employ {
    name: String,
    age: u32,
    sex: String,
}

fn main() {
    let mut em: Employ = Employ {
        name: String::from("000"),
        age: 30,
        sex: String::from("a"),
    };
    em.name = String::from("donglietao");
    println!("{}", em.name);
}

```

00000000

```

fn main() {
    let em = get_employ();
    // 0000

```

```

    let em2 = Employ { ..em };
    println!("{}", em.name);
    println!("{}", em2.name);

// 打印
    println!("{}", em2);
}

fn get_employ() -> Employ {
    let mut em: Employ = Employ {
        name: String::from("张三"),
        age: 30,
        sex: String::from("a"),
    };
    em.name = String::from("donglietao");
    return em;
}

// 打印
#[derive(Debug)]
struct Employ {
    name: String,
    age: u32,
    sex: String,
}

```

打印

```

fn main() {
    let em = Employ::new(String::from("张三"), 33, String::from("1"));
    println!("{}", em.to_string());
}

// 打印
#[derive(Debug)]
struct Employ {
    name: String,
    age: u32,
    sex: String,
}

```

```

}

// 实现
impl Employ {
    // 构造函数
    fn new(name: String, age: u32, sex: String) -> Employ {
        return Employ {
            name: name,
            age: age,
            sex: sex,
        };
    }

    // 打印
    // 打印self
    fn to_string(self) -> String {
        return self.name;
    }
}

```

枚举

```

enum Sex {
    Main = 0,
    Woman = 1,
}

```

#9

2023 14:40:48

2024 20:47:01