C：

1. 每一次调用函数都是一次拷贝形参
2. 对于变量的用另一个变量去赋值 都是默认属于拷贝
3. 定义变量 用 变量类型 变量名的方式定义： int my\_value;
4. 定义的变量默认可以修改值大小， 用const 来使得变量值不可变
5. 只有语句没有表达式的说法，所有的代码都是有分号作为结尾符？

Rust：

1. 每一次调用函数是所有权的移动，导致原来的变量不能再使用 ， 需要使用&来引用从而保证原来的变量可以继续使用。
2. 对于变量的用另一个变量去赋值 都是默认属于所有权的移动 导致原变量无法使用
3. 定义变量 用 let + 变量名: 变量类型 的方式定义： let my\_value: i32 ;
4. 定义的变量默认不可以修改值的大小， 用mut 来使得变量值可以改变
5. 有表达式的概念 即不使用分号作为结尾则自动代表赋值 例如：

let my\_value = {

let x = 5;

x + 1

}

1. 有一种特殊的元组的概念，从最开始定义了变量之后就不在产生改变，例如定义月份数组就可以使用元组的方式来定义。
2. 打印输出函数println!()，本身是使用了引用的方式来打印的，但是如果使用dbg！（），是转移了所有权的，注意首先使用#【derive（Debug）】，如果不想转移所有权还需要使用&来加入语句中。另一种使用 Debug 格式打印数值的方法是使用 [dbg! 宏](https://doc.rust-lang.org/std/macro.dbg.html)。dbg! 宏接收一个表达式的所有权（与 println! 宏相反，后者接收的是引用），打印出代码中调用 dbg! 宏时所在的文件和行号，以及该表达式的结果值，并返回该值的所有权。
3. 结构体不需要加分号， 初始化结构体的时候可以直接在语句中初始化， 不确定有没有构造函数。 关于结构体的输出方式可以使用Debug的方式来展现。
4. 新的变量类型， 枚举enum IpAddrKind {V4,V6,}
5. 利用枚举和match的方法来达到类似C语言中switch的效果。同时match也可以用来编写一个利用option匹配的函数来达到对应的数据自增或自减，没有所需要的数据就返回报错信息。在处理option中必须要有一个None的情况需要进行处理。
6. 同C语言if、else if，Else一样match从第一行语句开始检查，当遇到对应的分支的时候进入语句，后面语句将不在进行比较。
7. 在match中other的使用类似与else，但是同时这个other的值是可以直接在other语句中进行使用的。同样other必须放置在match语句中的最后一句。如果是不需要的数据就可以使用\_来进行通用处理。
8. 可以使用if let的方式来处理只用一种想要的match的情况。
9. 库，包，路径 暂时跳过

15.使用vector来存储列表。 例如let v: Vec<i32> = Vec::new();

更新vector使用push方法

读取vector的元素使用索引或者get方法都是可以的，当考虑有可能访问到数组外的数据时选择使用get方法来保证程序能够正常执行，返回None

遍历vector的方法是类似与py的写法 如下：

for i in &v

如果需要遍历vector同时对每一个元素进行改变需要用到可变引用的方式来解决。

For i in &mut v{ \*i += 50;}

其中需要注意使用\*来进行解引用。

使用枚举可以用来实现结构体数组。

1. 字符串集合string 例如：let mut s = String::new();

同时也可以使用如下方式来建立：

Let s = data.to\_string(); let s = String::from("initial contents");

可以使用push\_str方法来给字符串增加长度或者直接使用+

遍历字符串的方式使用的是

for c in “所需要的字符串”.char()

另外可以使用bytes()方法来返回原始字节

16.泛型