Introduction to Rust Programming

Hitesh Kumar Sharma
Instructor, Pluralsight

DR. HITESH KUMAR SHARMA

Technical Instructor & Consultant

ABOUT ME

- Industry/Academic Experience: **15 Years (8 years Exp in Rust, Kotlin, Python and Java Programming, 4 years in DevOps & Cloud Computing ,3 Years in Data Analytics)**
- Worked as **IBM Instructor**
- Worked as **Microsoft Instructor**
- **Working as PluralSight Instructor**
- Core Technical Domains: **Rust, Python, Java, DevOps, Cloud Computing, Data Analytics**
- Academic Qualifications**: Ph.D. (CSE), M.Tech (CSE)**
- Certifications:
  - **Confluent Developer, Admin Certified**
  - **UiPath RPA Certified Associate**
  - **Docker Certified Associate**
  - **Neo4J Certified Associate**
  - **Maven Certified Professional**
- 5 Books Published
- 35 Patents Published
- 02 Copyright Published

Agenda

- Introduction to Rust
- Rust Memory Model

Introduction to Rust Programming

- The Mozilla Corporation created the modern systems programming language called Rust.
- It is designed to be a language for extremely secure and concurrent systems.
- It is lightning fast like C and C++ since it compiles to native code.

Why Rust?

There are several reasons why programmers Favor Rust. The following are the causes:

- **Rust is Fast :** Rust code compiles to native machine code on several systems. This is the reason Rust is faster compare to other languages.
- **Rust is Memory Safe:** Rust encourages programmers to create secure programs Unlike C, it does not support dangling, uninitialized, and NULL pointers.
- **Rust is Low-Overhead:** Every value in the Rust programming language has a distinct owner, and the scope of the value matches the scope of the owner. It has an ownership system as a result.

Why Rust? (Contd..)

- **Rust is easy to use:** The syntax of the Rust programming language is comparable to that of C/C++, making it simple to use or comprehend.

- **Rust is statically and strongly typed:** Because of the way Rust is designed, code may be checked at compile time without any additional memory usage if the compilation fails.

- **Binding with C programs:** Similar to vectors, Rust offers a C API with memory safety that uses high-level functions.

- **Threads without Data race:** Data race is a condition where two or more threads access shared memory. Because of clear ownership rules, this condition does not arise in Rust.

Disadvantages of Rust Programming

- Rust may take more time to understand due to its complexity.
- Rust code has the potential to be less effective, and it also takes longer to compile.
- As applications developed in Rust are more complex, they may take longer to execute.
- Cyclical referencing an cause memory leaks, making the program execution slower.
- Due to its extensive code base, it is difficult to maintain.

Mutability in Rust

Values in rust are immutable by default and must be tagged as being mutable(if needed).

```
let x = 2;

x = 9; //it will show an error
```

The above example will show an error because we have not tagged it as mutable.

```
let mut x = 2;

x = 9; //work correctly
```

This will work fine as we have tagged it as being mutable.

Rust Type System

Every variable, value, and thing in Rust has a type. The type specifies which operations can be carried out on the value and how much memory will get allocated.

Integer Types in Rust

Length	Signed	Unsigned
8-bit	i8	u8
16-bit	i16	u16
32-bit	i32	u32
64-bit	i64	u64
128-bit	i128	u128
arch	isize	usize

Tuple in Rust

Tuples hold many values of different types, concurrently. Once a tuple is defined, it is immutable and there is implicit way to add/remove elements in a tuple. You can access a tuple's values using index. Tuples in Rust do not support iteration through loops.

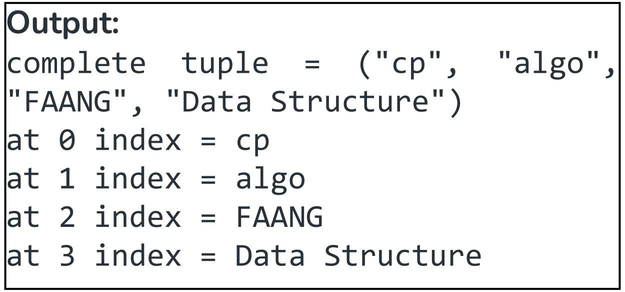
Use parenthesis to define a tuple as shown below.

Syntax: (“pluralsight", 1, plural')

Tuple in Rust (Example)

Example:

```
// Rust program to get value from tuple
// using index
fn main() {
		let ps = ("cp", "algo", "FAANG", "Data Structure");

		// complete tuple
		println!("complete tuple = {:?} ", ps );

		// first value
		println!("at 0 index = {} ", ps.0 );

		// second value
		println!("at 1 index = {} ", ps.1 );

		// third value
		println!("at 2 index = {} ", ps.2 );

		// fourth value
		println!("at 3 index = {} ", ps.3 );
}
```



Output:

```
complete tuple = ("cp", "algo",
"FAANG", "Data Structure")
at 0 index = cp
at 1 index = algo
at 2 index = FAANG
at 3 index = Data Structure
```

Structure in Rust

Rust uses the struct(ure) user-defined type to aggregate data elements of different types. Data is described by the structure as a key-value pair.

Syntax:

```
struct Name_of_structure
{
field1:data_type,
field2:data_type,
field3:data_type
}
```

Structure in Rust (Example)

Example:

```
struct Employee {
name: String,
company: String,
employee_id: u32,
profile: String
}
fn main() {
let value = Employee {
                    name: String::from(“PluralSight"),
                    company: String::from("pluralsight.com"),
                    employee_id: 007,
                    profile:String::from("Manager"),

};
println!("Employee {}: {} is a {} at {}.",
value.employee_id,
value.name,
value.profile,
value.company);

}
```

End of Module