Introduction

Why learn Rust programming language?

Rust is a memory-safe compiled-language and used inside proteinpaint to perform compute intensive calculations. The basic motto of Rust is to “Catch software bugs even before you make them” (e.g accessing variables that may be undefined in certain cases , data races). It does this by giving more compilation errors (while converting source code to binary) instead of runtime errors. This helps in reducing chance of code failing in a production server and increases chance of bugs getting caught in the developmental phase of the code itself.

This is accomplished by using the novel concept of ownership and borrowing of variables. For more details, please read the official [rust documentation](https://doc.rust-lang.org/book/ch04-00-understanding-ownership.html). This negates the use of a garbage-collector which helps in reducing execution time but at the same time makes it much more memory-safe than javascript and python.

For installing the rust compiler, please see [this](https://doc.rust-lang.org/cargo/getting-started/installation.html) documentation.

Problem: In the code below a struct employee is created which is copied (“apparently”) to newEmployee. Both variables are printed then the ename in newEmployee is changed to “Beck” and after that both variables are printed again.

Nodejs implementation

|  |
| --- |
| // Syntax: node shallow.js  const employee = {  eid: "E102",  ename: "Jack",  eaddress: "New York",  salary: 50000,  };    console.log("Employee=> ", employee);  const newEmployee = employee;  console.log("New Employee=> ", newEmployee);  console.log("---------After modification----------");  newEmployee.ename = "Beck";  console.log("Employee=> ", employee);  console.log("New Employee=> ", newEmployee); |

Save above code as shallow.js and execute the code as described in line 1.

Q1 Is there a discrepancy in the output? If yes, why is this discrepancy caused?

Ans: Yes, there is a discrepancy in the output as the “ename” value of “employee” object has also been updated to “Beck” instead of displaying the value as “Jack”.

The discrepancy is caused because when “employee” object is assigned to variable “newEmployee”, this is creating reference to original “employee” object instead of creating a new object in the memory since objects are passed by reference in Java script. Hence when the “ename” property is updated, it is changing values for both “employee” and “newEmployee” as well.

Rust implementation

|  |
| --- |
| // Syntax: rustc shallow.rs -o shallow && ./shallow    #[derive(Debug, Clone)]  struct Employee {  eid: String,  ename: String,  eaddress: String,  salary: i64,  }  fn main() {  let employee = Employee {  eid: String::from("E102"),  ename: String::from("Jack"),  eaddress: String::from("New York"),  salary: i64::from(50000),  };  println!("Employee=>{:?}", employee);  #[allow(non\_snake\_case)]  let mut newEmployee = employee;  println!("newEmployee=>{:?}", newEmployee);  newEmployee.ename = String::from("Beck"); // New value is assigned  println!("---------After modification----------");  println!("Employee=>{:?}", employee);  println!("newEmployee=>{:?}", newEmployee);  } |

Save the above code in a file as shallow.rs and compile and execute the code as described below:

rustc shallow.rs -o shallow && ./shallow

Q2 Does this code compile using the syntax above ? If not, why not (explain in words why it does not work)?

Ans : No, it does not compile and following is the error that was displayed.

Text

Description automatically generated

As per the above error remarks, the main() function present in line 9 is having an open parenthesis indicating the compiler that function body has started so now the complier is looking for a closing parenthesis from line 9 to 24 to identify where the function body ends. Since the closing parenthesis is not found even by the end of file, it is throwing an error that delimiter started at main() function is not closed.

Q3 Edit this code so that it compiles and gives the correct output?

Ans: Following is the edited code.

Output:

Text

Description automatically generated

Text

Description automatically generated