AMRITA VISHWA VIDYAPEETHAM

PRINCIPLES OF PROGRAMMING LANGUAGE LAB-7

Name: P Mokshagna Bhuvan **Course Code:**20CYS312

Roll no: CH.EN.U4CYS22041 **Date:** 21-02-25

Github link: https://github.com/MokshagnaBhuvan/myscript/tree/main/rust

Objective:

Rust is a powerful systems programming language known for its safety and performance. It provides strict type checking, ensuring that variables are correctly assigned and used. Learning Rust involves understanding its unique ownership and borrowing system. Conditional statements like `if-else` help in decision-making within programs. Loops such as `for` and `while` allow repeated execution of code efficiently.

1. Title: Data Types and Variables

Declare variables of the following types: integer, floating-point, boolean, and character. Print the value of each variable

Program:

```
fn main() {
    let int_var: i32 = 10; // Integer
    let float_var: f64 = 3.14; // Floating-point number
    let bool_var: bool = true; // Boolean
    let char_var: char = 'A'; // Character
    println!("Integer: {}", int_var);
    println!("Float: {}", float_var);
    println!("Boolean: {}", bool_var);
    println!("Character: {}", char_var);
}
```

Explanation:

We declare variables using the `let` keyword.Rust is statically typed, so we specify the type explicitly. We use `println!` macro to print variable values.

Output:

```
asecomputerlab@sanjay-21159:~$ nano t1.rs
asecomputerlab@sanjay-21159:~$ rustc t1.rs
asecomputerlab@sanjay-21159:~$ ./t1
Integer: 10
Float: 3.14
Boolean: true
Character: A
```

2. Title: Simple Arithmetic Operations

Declare two integer variables and perform the following operations:

Addition ,Subtraction, Multiplication, Division, Modulo

Program:

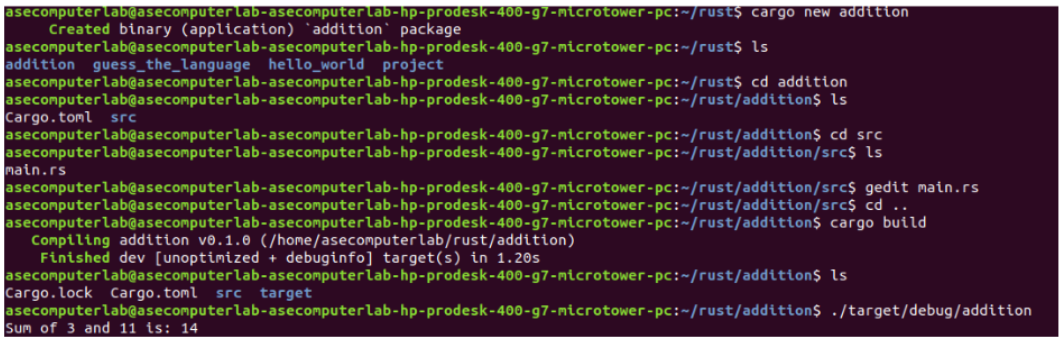
```
fn main() {
let x:f32 = 3.0;
let y:f32 = 11.0;
let sum:f32 = x+y;
println!("Sum of {x} and {y} is: {sum}");
} #addition

fn main() {
let x:u32 = 14;
let y:u32 = 3;
let diff:u32;
if x>y {
    diff = x-y;
} else {
       diff = y-x;
}

   println!("Difference of these {x} and {y} is {diff}");
} //subtraction


fn main() {
   let x:f32 = 3.0;
   let y:f32 = 12.0;
   let mul:f32 = x*y;
   println!("multiplication of {x} with {y} results: {mul}");
```

```
} //multiplication

fn main() {
    let x:f64 = 13.0;
   let y:f64 = 4.0;
   let div:f64 = x/y;
   println!("Division of {x} by {y} is: {div}");
} //division

fn main() {
   let x:f32 = 13.0;
   let y:f32 = 4.0;
   let modulo:f32 = x%y;
   println!("Modulo of {x} with {y} is: {modulo}");
} //modulo
```

Explanation:

Rust provides a simple and efficient way to perform arithmetic operations like addition, subtraction, multiplication, division, and modulo. Using variables with specific data types like f32, f64, and u32, Rust ensures type safety while performing calculations. The println! macro allows formatted output, making it easy to display results dynamically. With Rust's strict type system and memory safety features, arithmetic operations are both precise and reliable.

Output:



```
asecomputerlab@asecomputerlab-asecomputerlab-hp-prodesk-400-g7-microtower-pc:~/rust/addition$ cd ..
asecomputerlab@asecomputerlab-asecomputerlab-hp-prodesk-400-g7-microtower-pc:~/rust$ cargo new subtraction
     Created binary (application) `subtraction` package
asecomputerlab@asecomputerlab-asecomputerlab-hp-prodesk-400-g7-microtower-pc:~/rust$ cd subtraction/
asecomputerlab@asecomputerlab-asecomputerlab-hp-prodesk-400-g7-microtower-pc:~/rust/subtraction$ ls
Cargo.toml  src
asecomputerlab@asecomputerlab-asecomputerlab-hp-prodesk-400-g7-microtower-pc:~/rust/subtraction$ gedit src/main.rs
asecomputerlab@asecomputerlab-asecomputerlab-hp-prodesk-400-g7-microtower-pc:~/rust/subtraction$ cargo build
   Compiling subtraction v0.1.0 (/home/asecomputerlab/rust/subtraction)
error[E0425]: cannot find value `diff` in this scope
  --> src/main.rs:10:51
   |
10 |     println!("Difference of these {x} and {y} is {diff}");
   |                                                   ^^^^ not found in this scope

For more information about this error, try `rustc --explain E0425`.
error: could not compile `subtraction` due to previous error
asecomputerlab@asecomputerlab-asecomputerlab-hp-prodesk-400-g7-microtower-pc:~/rust/subtraction$ gedit src/main.rs
asecomputerlab@asecomputerlab-asecomputerlab-hp-prodesk-400-g7-microtower-pc:~/rust/subtraction$ cargo build
   Compiling subtraction v0.1.0 (/home/asecomputerlab/rust/subtraction)
    Finished dev [unoptimized + debuginfo] target(s) in 0.17s
asecomputerlab@asecomputerlab-asecomputerlab-hp-prodesk-400-g7-microtower-pc:~/rust/subtraction$ ./target/debug/subtraction
Difference of these 14 and 3 is 11
```

```
asecomputerlab@asecomputerlab-asecomputerlab-hp-prodesk-400-g7-microtower-pc:~/rust$ cargo new multiplication
     Created binary (application) `multiplication` package
asecomputerlab@asecomputerlab-asecomputerlab-hp-prodesk-400-g7-microtower-pc:~/rust$ cd multiplication/; gedit src/main.rs
asecomputerlab@asecomputerlab-asecomputerlab-hp-prodesk-400-g7-microtower-pc:~/rust/multiplication$ cd multiplication/; gedit src/main.rs
bash: cd: multiplication/: No such file or directory
asecomputerlab@asecomputerlab-asecomputerlab-hp-prodesk-400-g7-microtower-pc:~/rust/multiplication$ ./target/debug/multiplication
bash: ./target/debug/multiplication: No such file or directory
asecomputerlab@asecomputerlab-asecomputerlab-hp-prodesk-400-g7-microtower-pc:~/rust/multiplication$ cargo build
   Compiling multiplication v0.1.0 (/home/asecomputerlab/rust/multiplication)
    Finished dev [unoptimized + debuginfo] target(s) in 0.20s
asecomputerlab@asecomputerlab-asecomputerlab-hp-prodesk-400-g7-microtower-pc:~/rust/multiplication$ ./target/debug/multiplication
multiplication of 3 with 12 results: 36
```

```
asecomputerlab@asecomputerlab-asecomputerlab-hp-prodesk-400-g7-microtower-pc:~/rust/division$ cd ..;cargo new modulo
     Created binary (application) `modulo` package
asecomputerlab@asecomputerlab-asecomputerlab-hp-prodesk-400-g7-microtower-pc:~/rust$ cd modulo;gedit src/main.rs
asecomputerlab@asecomputerlab-asecomputerlab-hp-prodesk-400-g7-microtower-pc:~/rust/modulo$ cargo build
   Compiling modulo v0.1.0 (/home/asecomputerlab/rust/modulo)
    Finished dev [unoptimized + debuginfo] target(s) in 0.21s
asecomputerlab@asecomputerlab-asecomputerlab-hp-prodesk-400-g7-microtower-pc:~/rust/modulo$ ./target/debug/modulo
Modulo of 13 with 4 is: 1
asecomputerlab@asecomputerlab-asecomputerlab-hp-prodesk-400-g7-microtower-pc:~/rust/division$ ./target/debug/division
Division of 13 by 4 is: 3.25
```

3. Title: If-Else Decision Making

Write a program that:

Takes a number as input.

Checks whether the number is positive, negative, or zero using an if-else statement.

Print a message based on the result.

Program:

```
use std::io;


fn main() {
   let mut input = String::new();
   io::stdin().read_line(&mut input).expect("Failed to read line");

   let x: i32 = input.trim().parse().expect("Please enter a valid integer");
```

```
  if x > 0 {
     print!("{x} is positive.");
  } else if x == 0 {
     print!("{x} is Zero.");
  } else {
     print!("{x} is negative.");
  }
} //if- else
```

Explanation:

This Rust program reads an integer input from the user and determines whether it is positive, negative, or zero. The `read_line()` function reads the input, and `.trim().parse()` converts it to an integer. Using an `if-else` statement, it checks the sign of the number and prints the appropriate message. This ensures user-friendly output based on the given number.

Output:

```
asecomputerlab@asecomputerlab-asecomputerlab-hp-prodesk-400-g7-microtower-pc:~/rust/if-else$ cd if-else/;gedit src/main.rs
bash: cd: if-else/: No such file or directory
asecomputerlab@asecomputerlab-asecomputerlab-hp-prodesk-400-g7-microtower-pc:~/rust/if-else$ cargo build
   Compiling if-else v0.1.0 (/home/asecomputerlab/rust/if-else)
    Finished dev [unoptimized + debuginfo] target(s) in 0.28s
asecomputerlab@asecomputerlab-asecomputerlab-hp-prodesk-400-g7-microtower-pc:~/rust/if-else$ ./target/debug/if-else
100
100 is positive.asecomputerlab@asecomputerlab-asecomputerlab-hp-prodesk-400-g7-microtower-pc:~/rust/if-else$
```

4. Title: Checking for Even or Odd

Write a program that: Takes an integer as input.Uses an if-else statement to check if the number is even or odd. Print "Even" if the number is even and "Odd" if the number is odd.

Program:

```
fn main() {

  let mut input = String::new();
  io::stdin().read_line(&mut input).expect("Failed to read line");

  let x: i32 = input.trim().parse().expect("Please enter a valid integer");
  if (x%2)==0 {
     print!("{x} is Even.");
  } else {
          print!("{x} is Odd.");
  }

} //odd or even
```

Explanation:

This program reads an integer from the user and determines if it is odd or even. The input is taken as a string and converted into an integer using `.parse()`. It then checks if the number is divisible by 2 using `x % 2 == 0`. If true, it prints that the number is even; otherwise, it prints that the number is odd.

Output:

```
asecomputerlab@asecomputerlab-asecomputerlab-hp-prodesk-400-g7-microtower-pc:~/rust/printEven$ gedit src/main.rs
asecomputerlab@asecomputerlab-asecomputerlab-hp-prodesk-400-g7-microtower-pc:~/rust/printEven$ cargo build
   Compiling printEven v0.1.0 (/home/asecomputerlab/rust/printEven)
warning: crate `printEven` should have a snake case name
  |
  = note: `#[warn(non_snake_case)]` on by default
  = help: convert the identifier to snake case: `print_even`

warning: `printEven` (bin "printEven") generated 1 warning
    Finished dev [unoptimized + debuginfo] target(s) in 0.23s
asecomputerlab@asecomputerlab-asecomputerlab-hp-prodesk-400-g7-microtower-pc:~/rust/printEven$ ./target/debug/printEven
15
15 is Odd.asecomputerlab@asecomputerlab-asecomputerlab-hp-prodesk-400-g7-microtower-pc:~/rust/printEven$ ./target/debug/printEven
16
16 is Even.asecomputerlab@asecomputerlab-asecomputerlab-hp-prodesk-400-g7-microtower-pc:~/rust/printEven$
```

5. Title: Using a Loop to Print Numbers

Write a program that uses a for loop to print the even numbers from the range 1 to 20.

Program:

```
use std::io;

fn main() {
  //let mut input = String::new();
  //io::stdin().read_line(&mut input).expect("Failed to read line");

  let x: i32 ;// = input.trim().parse().expect("Please enter a valid integer");
  for x in 1..20{
    if (x%2)==0 {
          print!("{x} is Even.");
    }
  }

}// print even
```

Explanation:

The program iterates through numbers from 1 to 19 using a `for` loop. It checks whether each number is even by using the condition `x % 2 == 0`. If the condition is met, the program prints that the number is even. However, there's an issue: the `x` variable is declared but not assigned, which might cause an error.

Ouput:

```
is is Even.asecomputerlab@asecomputerlab-asecomputerlab-hp-prodesk-400-g7-microtower-pc:~/rust/printEven$ gedit src/main.rs
asecomputerlab@asecomputerlab-asecomputerlab-hp-prodesk-400-g7-microtower-pc:~/rust/printEven$ cargo build
   Compiling printEven v0.1.0 (/home/asecomputerlab/rust/printEven)
warning: unused import: `std::io`
 --> src/main.rs:1:5
  |
1 | use std::io;
  |     ^^^^^^^
  |
  = note: `#[warn(unused_imports)]` on by default

warning: unused variable: `x`
 --> src/main.rs:6:9
  |
6 |     let x: i32 ;// = input.trim().parse().expect("Please enter a valid integer");
  |         ^ help: if this is intentional, prefix it with an underscore: `_x`
  |
  = note: `#[warn(unused_variables)]` on by default

warning: crate `printEven` should have a snake case name
  |
  = note: `#[warn(non_snake_case)]` on by default
  = help: convert the identifier to snake case: `print_even`

warning: `printEven` (bin "printEven") generated 3 warnings
    Finished dev [unoptimized + debuginfo] target(s) in 0.29s
asecomputerlab@asecomputerlab-asecomputerlab-hp-prodesk-400-g7-microtower-pc:~/rust/printEven$ ./target/debug/printEven
2 is Even.4 is Even.6 is Even.8 is Even.10 is Even.12 is Even.14 is Even.16 is Even.18 is Even.asecomputerlab@asecomputerlab-asecomputerlab-hp-prodesk-400-g7-microtower-pc:~/ru
```

6. Title: While Loop Example

Write a program that uses a while loop to print odd numbers from the range 1 to 20.

Code:

```
fn main() {
   let mut x = 1;

   while x < 20 {
      if x % 2 != 0 {
         println!("{x}.");
      }
      x += 1;
   }
} //print odd
```

Explanation:

This program uses a `while` loop to iterate from 1 to 19. Inside the loop, it checks whether a number is odd using `x % 2 != 0`. If true, it prints the number, then increments `x` by 1. This ensures that all odd numbers in the range are displayed correctly.

Output:

```
asecomputerlab@asecomputerlab-asecomputerlab-hp-prodesk-400-g7-microtower-pc:~/rust/whileloop$ cd whileloop/;gedit src/main.rs
bash: cd: whileloop/: No such file or directory
asecomputerlab@asecomputerlab-asecomputerlab-hp-prodesk-400-g7-microtower-pc:~/rust/whileloop$ cargo build
   Compiling whileloop v0.1.0 (/home/asecomputerlab/rust/whileloop)
    Finished dev [unoptimized + debuginfo] target(s) in 0.18s
asecomputerlab@asecomputerlab-asecomputerlab-hp-prodesk-400-g7-microtower-pc:~/rust/whileloop$ ./target/debug/whileloop
1.
3.
5.
7.
9.
11.
13.
15.
17.
19.
asecomputerlab@asecomputerlab-asecomputerlab-hp-prodesk-400-g7-microtower-pc:~/rust/whileloop$
```

7. Title : Using a For Loop with a Range .

Write a program that uses a for loop to print the numbers from 10 to 1 in reverse order (10, 9, 8, ..., 1).

Code:

```
fn main(){
   for num in (1..10).rev() {
      println!("{num}");
   }
} // rev 10
```

Explanation:

This program prints numbers from 10 to 1 using a `for` loop with the `.rev()` function. The range `(1..10).rev()` generates numbers from 1 to 9 in reverse order. The `println!` macro then prints each number, effectively creating a countdown sequence.

Output:

Conclusion:

Rust enforces strong typing, preventing many common runtime errors. Arithmetic operations follow familiar rules, making mathematical calculations straightforward. Conditional logic is similar to other languages, ensuring an easy learning curve. Loops enable iteration over ranges and collections effectively. Rust's memory safety without garbage collection makes it a reliable choice for high-performance applications.