**FIBONACCI SERIES GENERATOR**

By

Adewuyi Goodness Adeyinka

Presented to:

The Faculty of Computing and Informatics

Department of Computer Science

Ladoke Akintola University of Technology

200 Level SWEP Project

Under the supervision of:

Prof. O.O. Adeosun and Mrs. Jeremiah Y.S.

February 27, 2024

**Table of Contents**

**Chapter One …………………………………………… 1**

Introduction ………………………………..……………. 1

**Chapter Two …………………………………………… 3**

History of Rust Programming Language

**Chapter Three …………………………………………. 5**

Objectives …………………………………………. 5

Features ……………………………………………. 5

**Chapter Four …………………………………………… 6**

Methodology …………………………………………….. 6

1. Define Constants ………………………………….. 6
2. Initialize variables …………………………………. 6
3. Creating Conditional statements
4. Loops
5. Combine steps

**Chapter Five ……………………………………………..**

Result

Conclusion

**Chapter One**

**Introduction:**

Welcome to the exciting world of Rust programming! In this project, we will delve into the realm of Fibonacci series using the Rust programming language. The Fibonacci sequence is a fascinating mathematical concept where each number is the sum of the two preceding ones, typically starting with 0 and 1.

By exploring this sequence in Rust, we will not only showcase the power and elegance of Rust's syntax but also demonstrate its efficiency in handling complex algorithms. Join me in this journey as I unravel the beauty of Fibonacci numbers through the lens of Rust programming. Let's dive in and discover the magic of Fibonacci series in Rust!

**Chapter Two**

**History of Rust Programming Language**

**Origins and Development:**

Rust was created as a personal project by Graydon Hoare in 2006 while he was working at Mozilla. The language was officially announced in 2010, and its first stable version, Rust 1.0, was released in 2015. Initially, Rust faced criticism for its complexity, steep learning curve, verbosity, and limited ecosystem. However, over time, these issues were addressed, leading to significant improvements in the language.

**Key Milestones:**

Through the early 2010s, Mozilla engineers and Rust volunteers refined Rust's core, focusing on memory management and creating an "ownership" system to reduce memory problems. By 2013, Rust had removed its garbage collector, enhancing performance and efficiency while maintaining memory safety.

**Community and Popularity:**

Rust has gained momentum and popularity over the years, with increasing visibility in various domains such as the Linux Kernel and support from big corporations. The Rust community is known for its friendliness and openness to newcomers, fostering a welcoming environment for all contributors.

**Name Origin:**

The name "Rust" is derived from a fungus known for being robust, distributed, and parallel. It also reflects the language's focus on avoiding new technology and incorporating well-established concepts.

This brief history encapsulates the evolution of Rust from its inception to becoming one of the most beloved programming languages today.

**Chapter Four**

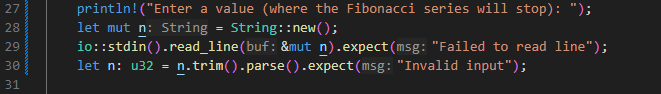
**Methodology:**

A methodological approach to writing a program to find the Fibonacci series using the Rust programming language follows these steps.

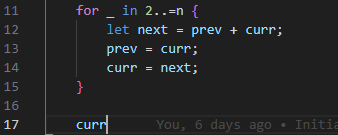
1. **Initialize variables:** Declare two mutable variables that stores the two initial starting values for the Fibonacci series, 0 and 1.



Declare another variable that stores where you want the Fibonacci series to be stopped. This value is supplied by the user.



1. Calculate the next value from the two initial values of the Fibonacci series.



1. **Output the result:** Display the Fibonacci series, in this case to the 10th term.

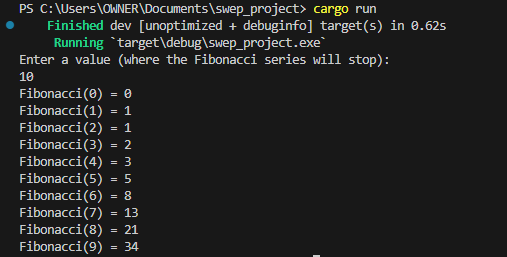
swep_project_004

**Chapter Five**

**Result:**

The result of our code gives the following results:

In this scenario, the user enters 10 as where the series should stop.



**Conclusion:**

In conclusion, this project has been a journey through the elegant and powerful world of Rust programming, exploring the timeless concept of the Fibonacci series. By leveraging Rust's robust features and syntax, we have delved into the intricacies of generating Fibonacci numbers efficiently and accurately.

Throughout this project, we have witnessed how Rust's memory safety, concurrency support, and performance optimizations make it an ideal choice for handling complex algorithms like the Fibonacci sequence. By combining mathematical concepts with Rust's capabilities, we have not only showcased the beauty of Fibonacci numbers but also demonstrated the prowess of Rust as a modern programming language.