20CYS312 - Principles of Programming Languages
Exploring Programming Paradigms

Assignment-01

Presented by M K Ashwatha Prasad
CB.EN.U4CYS21036
TIFAC-CORE in Cyber Security
Amrita Vishwa Vidyapeetham, Coimbatore Campus

Feb 2024

Table Of Contents

Rust

Introduction to Rust:
Rust is a modern systems programming language designed for performance, safety, and concurrency. Developed by Mozilla, Rust aims to provide a reliable and efficient alternative to languages like C and C++ while addressing common pitfalls related to memory safety and concurrency issues.

Key Features of rust:

- Pattern Matching
- Ownership
- Zero cost Abstractions
- Concurrency
- Error-Handling

Features

Pattern Matching Rust supports pattern matching through the match keyword. This allows developers to destructure data and match it against specific patterns, making code more expressive and readable. Pattern matching is particularly powerful when working with enums.

Ownership Rust's ownership system ensures that each piece of memory has a single owner. This owner is responsible for cleaning up the memory when it's no longer needed. This prevents issues like memory leaks.

Zero Cost Abstraction Rust's philosophy of "zero-cost abstractions" means that high-level language features don't come with a runtime performance penalty.

Concurrency Rust's ownership system plays a crucial role in preventing data races in concurrent programming. The ownership rules ensure that multiple threads cannot simultaneously modify the same piece of data, avoiding common pitfalls associated with shared mutable state.

Error Handling Rust uses the Result type for error handling. Functions that can produce errors return a Result with either a value or an error. This approach enforces explicit error handling, reducing the likelihood of unchecked exceptions.

Expression in Languages

C:

- Memory Management: Rust's ownership system ensures memory safety without garbage collection. In C, manual memory management is common, similar to Rust. Developers need to allocate and deallocate memory explicitly using functions like malloc and free.

C++:

- Ownership and Borrowing: Rust's ownership model can be somewhat compared to C++'s smart pointers, like std::unique$_p trandstd ::$ $shared_p tr.However, Rust' sborrowcheckerenforcesstrictrulesatcompiletime, whileC + +reliesonmanualmemory$

Python:

- Memory Safety: Python uses automatic memory management (garbage collection), eliminating the need for manual memory management like in Rust. However, this comes at the cost of performance compared to low-level languages.

Java:

- Memory Safety: Similar to Python, Java uses automatic memory management through its garbage collector. Developers don't need to manage memory manually, as in Rust.

Real World Applications

Firefox:
Mozilla has been incorporating Rust into components of the Firefox web browser. Specifically, the browser engine, known as Servo, is built with Rust. Rust's memory safety features are particularly beneficial for critical components like browser engines.
Dropbox:
Dropbox has been exploring the use of Rust for performance-critical components. They have open-sourced projects like RustPython, a Python interpreter written in Rust. While this is a separate project from the main Dropbox application, it showcases the use of Rust in a real-world scenario.
Cloudflare
Cloudflare, a company providing internet security and performance services, has been adopting Rust for certain components. They have developed projects like quiche, which is a general-purpose QUIC library that's written in Rust and powers Cloudflare's QUIC and HTTP/3 implementations.

CHR

Introduction to CHR:
CHR, which stands for Constraint Handling Rules, is a declarative programming paradigm designed for expressing and solving constraint satisfaction problems (CSPs). It provides a high-level, rule-based language for specifying relationships between variables through constraints and allows for efficient constraint propagation.

Key Features:

- Declarative Rule-Based Language
- Guarded Heads
- Conditions and Actions
- Constraint Propagation

Features of CHR

Rule Based Programming:
CHR is centered around a rule-based programming paradigm. Programmers define rules that express relationships between variables through constraints.

Guarded Rules:
Rules in CHR have guarded heads, meaning that conditions (constraints) must be satisfied for the rule to be applied. This guards against applying rules in situations where their constraints are not met.

Conditions and actions:
Rules in CHR consist of conditions specifying constraints and actions defining changes to the constraint store when the rule is applied. This separation of conditions and actions enhances modularity.

Constraint Propogation
CHR is specifically tailored for solving Constraint Satisfaction Problems (CSPs), where the goal is to find values for a set of variables that satisfy a given set of constraints.

Real world applications

National language processing:
CHR has been explored in the context of natural language processing. Researchers have investigated its use in parsing and processing linguistic structures, leveraging the rule-based nature of CHR to express relationships and constraints in language understanding.

Bioinformatics:
Some research projects in bioinformatics have used CHR for expressing and solving constraint problems related to genomic data analysis. The rule-based approach of CHR can be suitable for specifying complex relationships in this domain.

Program analysis and verifications:
CHR has been applied in research related to program analysis and verification. It has been used to express and solve constraints arising from program properties, aiding in the development of tools for static analysis and verification of software.

Comparison

Memory management:

- **Rust:** Rust employs a unique ownership system that allows for safe memory management without garbage collection. It ensures memory safety by enforcing ownership, borrowing, and lifetimes at compile time.
- **CHR:** CHR, being a declarative language, abstracts away low-level memory management concerns. It focuses on expressing constraints and solving problems without manual memory management.

Concurrency:

- **Rust:** Rust has a strong focus on concurrency and provides ownership and borrowing mechanisms to ensure thread safety. It encourages writing concurrent code without data races.
- **CHR:** CHR does not inherently provide features for concurrency like Rust. However, its rule-based nature can be used to express constraints in concurrent systems.

Comparisons

Typing system:

- **Rust:** Rust has a static and strong typing system. It ensures type safety and prevents many common programming errors at compile time.
- **CHR:** CHR typically operates in a dynamically typed environment, and the constraints are often checked and resolved at runtime.

Application Domain:

- **Rust:** Rust is versatile and can be used for a wide range of applications, including systems programming, web development, and more.
- **CHR:** CHR is specialized for solving constraint satisfaction problems and is commonly used in domains where relationships between variables need to be expressed and efficiently solved.

References

https://en.wikipedia.org/wiki/Rust$_{(}programming_{l}anguage$)
https://rust-unofficial.github.io/patterns/functional/paradigms.html

https://blog.devgenius.io/exploring-rust-a-paradigm-shifting-programming-language-2193a100f830
https://en.wikipedia.org/wiki/Constraint$_{H}andling_{R}ules : \quad : text = Constraint$
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7704556/