20CYS312 - Principles of Programming Languages
Exploring Programming Paradigms

Assignment-01

Presented by Gokul
CB.EN.U4CYS21018
TIFAC-CORE in Cyber Security
Amrita Vishwa Vidyapeetham, Coimbatore Campus

Feb 2024

Outline

Imperative Paradigm

- Describes a sequence of steps for computer execution.
- Involves manipulating state variables to change the program state.
- Assumes the computer keeps track of state variables during computation.
- Order of steps is crucial, with consequences based on current variable values.
- One of the oldest paradigms, closely tied to Von Neumann architecture, emphasizing step-by-step tasks with limited abstraction.

Imperative - Rust

- Rust designed for diverse applications: systems, web, game development, and embedded systems.
- Multi-paradigm, general-purpose language prioritizing performance, type safety, and concurrency.
- Enforces memory safety without relying on automated memory management.
- Named after rust fungus, symbolizing robustness, distribution, and parallelism.
- Not confined to a single paradigm; supports imperative programming with safety benefits.
- Encourages immutability by default, facilitating safety in imperative programming.
- While requiring more effort, Rust code is safer than equivalent imperative code.

Hello World in Rust

```
// This is the main function.
fn main() {
    // Statements here are executed when the compiled binary is called.
    // Print text to the console.
    println!("Hello World!");
}

fn main() {
let n = 5;
if n < 0 {
print!("{} is negative", n);
} else if n > 0 {
print!("{} is positive", n);
} else {
print!("{} is zero", n);
}
```

Functional Paradigm

- Functional programming rooted in mathematics.
- Abstraction centers on functions for specific computations.
- Data and functions are independent; no global variable changes.
- Functions hide implementation details, allowing argument changes without altering meaning.
- Views subprograms as mathematical functions; solution depends on input, time is irrelevant.
- Does not manipulate state variables; follows immutability and no side effects.
- Focus on program goals rather than implementation details; emphasis on what, not how.

Functional - Scala

- Scala: modern, multi-paradigm language.
- Integrates object-oriented and functional features seamlessly.
- Every function treated as a value.
- Supports anonymous functions, higher-order functions, nesting, and currying.
- Utilizes case classes and pattern matching for algebraic types.
- Singleton objects for non-class functions, enhancing code efficiency.
- Supports a combination of functional and object-oriented programming styles.
- Versatile for building fast, concurrent, and distributed systems.
- Prioritizes interoperability, easy access to industry-proven libraries.
- Static types enhance safety with built-in checks and thread-safe structures.

Hello World in Scala

```
object Hello {
  def main(args: Array[String]) = {
    println("Hello, world")
  }
}

//Pure Functions

def sum(xs: List[Int]): Int = xs match
case Nil => 0
case head :: tail => head + sum(tail)
```

Imperative Paradigm:

- Sequences of steps for computer execution.
- Manipulates state variables to change program state.
- Order of steps crucial; consequences depend on variable values.
- Close relation to machine architecture (Von Neumann).
- Emphasizes how to achieve goals with little abstraction.

Functional Paradigm:

- Rooted in mathematics; focuses on functions.
- Data and functions decoupled; no global variable changes.
- Abstraction through functions hiding implementation details.
- Treats subprograms as mathematical functions, solution based on input.
- Emphasizes what the program should achieve; immutability and no side effects.

References

- "Introduction of Programming Paradigms" - GeeksforGeeks
  https://www.geeksforgeeks.org/introduction-of-programming-paradigms/
- "Major Programming Paradigms" - University of Central Florida
  https://www.cs.ucf.edu/leavens/ComS541Fall97/hw-pages/paradigms/major.htmlimperative
- "Rust Programming Language" - Wikipedia
  https://en.wikipedia.org/wiki/Rust(programminglanguage)
- "Rust Documentation" - Rust Programming Language
  https://doc.rust-lang.org/
- "Scala Programming Language" - Scala
  https://www.scala-lang.org/
- "Introduction to Scala" - Scala Documentation
  https://docs.scala-lang.org/scala3/book/introduction.html
- "Difference Between Functional and Imperative Programming" - GeeksforGeeks
  https://www.geeksforgeeks.org/difference-between-functional-and-imperative-programming/