20CYS312 - Principles of Programming Languages
Exploring Programming Paradigms

Assignment-01

Presented by K S SANTHOSSH
CB.EN.U4CYS21025
TIFAC-CORE in Cyber Security
Amrita Vishwa Vidyapeetham, Coimbatore Campus

Feb 2024

Outline

Programming Paradigm

1. Paradigm can also be termed as method to solve some problem or do some task.
2. It reflects a set of principles, concepts, and practices that programmers follow to structure their code and solve problems.
3. Different paradigms provide distinct ways to organize and structure code, determining how data is represented and manipulated.
4. Some programming languages are designed to support specific paradigms, while others are multi-paradigm, allowing developers to use multiple styles.
5. There are various programming paradigms. They are broadly divided in two sub classes. They are:
    1. Imperative programming paradigm
    2. Declarative programming paradigm

Functional Programming

Functional programming:

1. Functional programming is an approach to software development that uses pure functions to create maintainable software. In other words, building programs by applying and composing functions.
2. Its main focus is on "what to solve" in contrast to an imperative style where the main focus is "how to solve". It uses expressions instead of statements.
3. Functions are also treated like first-class citizens—meaning they can pass as arguments, return from other functions, and attach to names.
4. FP evolved from the lambda calculus (-calculus), a simple notation for functions and applications that mathematician Alonzo Church developed in the 1930s.

Functional Programming

Concepts of Functional Programming:

1. **First-Class Functions:** Functions are treated as first-class citizens, meaning they can be passed as arguments to other functions, returned as values from other functions, and assigned to variables.
2. **Pure Functions:** A pure function is a function that, given the same input, will always return the same output and has no observable side effects. It doesn't rely on or modify external state.
3. **Immutability:**
   Data is immutable, which means once it's created, it cannot be changed. Instead of modifying existing data structures, new ones are created.
4. **Referential Transparency:**
   Expressions or functions that can be replaced with their values without changing the program's behavior are said to be referentially transparent. This is closely related to the idea of pure functions.

Functional Programming:

1. **Higher-Order Functions:** Higher-order functions take one or more functions as arguments or return a function as a result. Functions are treated as first-class citizens.
2. **Recursion:** Loops are replaced with recursive function calls. Recursion is a fundamental concept in functional programming.
3. **Declarative Style:** The focus is on what the program should accomplish rather than how it should achieve it. This is in contrast to imperative programming, where the emphasis is on how to achieve a result.
4. **Pattern Matching:** A mechanism for checking a value against a pattern. It is a more concise way to write conditional statements.

Meta LAnguage

About:

1. ML is a general-purpose functional programming language developed in the 1970s at the University of Edinburgh.
2. ML has influenced the design of various programming languages, including Haskell and OCaml. The ML language family includes Standard ML (SML), OCaml, and others.
3. Key features of ML (the programming language) include strong static typing, type inference, pattern matching, and higher-order functions. It is often used in the development of compilers, theorem provers, and functional programming research.
4. ML is best thought as a mult-paradigm language: it supports both functional programming (through a full array of tools such as anonymous first-class functions) and imperative programming (through reference types and "functions" that have effects).

Meta Language

Some key aspects of MetaLanguage:

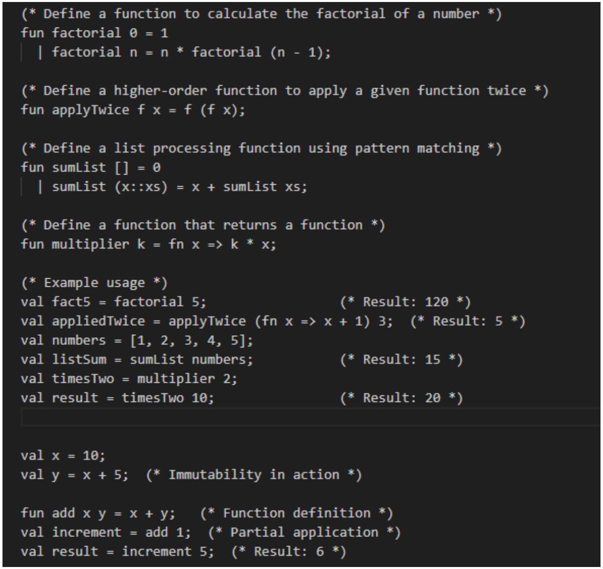
1. **Interactive language:** ML is an interactive language. Every phrase read is analyzed, compiled, and executed, and the value of the phrase is reported, together with its type.
2. **Abstract types:** ML supports abstract types. Abstract types are a useful mechanism forprogram modularization. New types together with a set of functions on objects of that type may be designed. The details of the implementation are hidden from the user of the type, achieving a degree of isolation that is crucial to program maintenance.
3. **Strong Typing:**
   ML is known for its strong, static type system. The type of a variable is checked at compile-time, and type inference is used to deduce types where possible. Strong typing helps catch errors early in the development process.
4. **Functional Programming:**
   ML is a functional programming language, emphasizing the use of functions as first-class citizens. Higher-order functions, closures, and recursion are integral to the language.

Meta Language

1. **Immutability:**
   ML encourages the use of immutable data structures. Once a value is assigned, it cannot be changed. Instead of modifying existing data structures, new ones are created.
2. **Garbage Collection:**
   Memory management in ML is handled through automatic garbage collection. This relieves developers from manual memory allocation and deallocation concerns.
3. **Polymorphism:**
   ML supports polymorphism, allowing the definition of generic functions and data types. Parametric polymorphism is achieved through type variables.
4. **Exception Handling:**
   Exception handling in ML provides a mechanism to gracefully handle errors and exceptional situations in a program.

Meta Language

Implementation:



```
(* Define a function to calculate the factorial of a number *)
fun factorial 0 = 1
  | factorial n = n * factorial (n - 1);

(* Define a higher-order function to apply a given function twice *)
fun applyTwice f x = f (f x);

(* Define a list processing function using pattern matching *)
fun sumList [] = 0
  | sumList (x::xs) = x + sumList xs;

(* Define a function that returns a function *)
fun multiplier k = fn x => k * x;

(* Example usage *)
val fact5 = factorial 5;                (* Result: 120 *)
val appliedTwice = applyTwice (fn x => x + 1) 3;  (* Result: 5 *)
val numbers = [1, 2, 3, 4, 5];
val listSum = sumList numbers;          (* Result: 15 *)
val timesTwo = multiplier 2;
val result = timesTwo 10;               (* Result: 20 *)


val x = 10;
val y = x + 5;  (* Immutability in action *)

fun add x y = x + y;   (* Function definition *)
val increment = add 1;  (* Partial application *)
val result = increment 5;  (* Result: 6 *)
```

Meta Language

1. The factorial| function demonstrates pattern matching and recursion.
2. The applyTwice| function shows the use of higher-order functions.
3. The sumList| function illustrates pattern matching on lists.
4. The multiplier| function returns a function as a result.
5. In ML, functions are first-class citizens, meaning they can be treated as values. You can assign functions to variables, pass them as arguments to other functions, and return them as results.
6. ML encourages the use of immutable data structures. Once a value is assigned, it cannot be changed. Instead of modifying existing data structures, new ones are created.(x and y).

Meta Language

Applications:
ML family of languages, such as Standard ML and OCaml. These languages have been applied in various domains due to their strong type systems, expressive features, and suitability for certain types of applications. Here are some applications of ML languages:

1. Compiler Construction
2. Theorem Proving and Formal Verification
3. Language Design and Implementation
4. Automated Reasoning and AI Research
5. Financial Modeling

Logical Programming

1. The logic programming paradigm is a programming paradigm that is based on formal logic.
2. It is characterized by expressing a program as a set of logical statements, and computation is carried out by applying inference rules to derive new logical statements.
3. It uses logic circuits to control how facts and rules about the problems within the system are represented or expressed.
4. logic is used to represent knowledge, and inference is used to manipulate it. It tells the model about how to accomplish a goal rather than what goal to accomplish.
5. Rules are written as logical clauses with a head and a body; for instance, "H is true if B1, B2, and B3 are true." Facts are similar to rules but without a body; for instance, "H is true."
6. Some logic programming languages, such as Datalog and ASP (Answer Set Programming), are purely declarative.

Logical Programming

Key Features:

1. **Declarative Nature:** Logic programming is declarative, meaning that the programmer specifies what needs to be done rather than explicitly describing how to do it.
2. **Rules and Facts:** Logic programs consist of rules and facts. Facts are statements that are assumed to be true, while rules define relationships and conditions. The inference engine uses these rules and facts to derive new conclusions.
3. **Inference:** The inference engine uses logical reasoning to derive conclusions from the given set of rules and facts. This process is often referred to as backward chaining, where the system starts with a goal and works backward to find a solution.
4. **Pattern Matching:** Logic programming languages typically involve pattern matching, where the system matches patterns in the input data against the rules and facts defined in the program.
5. **Non-deterministic Execution:** Logic programming languages often allow non-deterministic execution, where multiple solutions to a problem can be explored.
6. **Backtracking:** If a certain branch of execution does not lead to a solution, the logic programming system can backtrack and explore alternative paths.

ASP.NET

About:

1. .NET is a **web development** platform that assists you with the services required for building robust web applications and comprehensive software infrastructure.
2. ASP.NET extends the .NET platform with tools and libraries specifically for building web apps.
3. When using ASP.NET your back-end code, such as business logic and data access, is written using C#, F#, or Visual Basic.
4. ASP.NET is developed on the Common Language Runtime (CLR), thereby permitting the developers to build Web pages using any supported language.
5. The successor of ASP.NET is ASP.NET Core that is a re-implementation of ASP.NET in the form of a modular Web framework in combination with the Entity framework.

ASP.NET

Some key aspects:

1. **Model-View-Controller (MVC) Architecture:** Structured Development: ASP.NET MVC provides a model-view-controller architecture, promoting a structured and modular approach to building web applications.
2. **Web Forms:** ASP.NET Web Forms is an alternative model that follows an event-driven programming paradigm. It provides a more stateful and rapid application development approach with server controls.
3. **Integrated Development Environment (IDE):**ASP.NET development is often done using Microsoft Visual Studio, a powerful integrated development environment (IDE). Visual Studio offers extensive tools for designing, coding, testing, and debugging ASP.NET applications.
4. **Real-Time Communication:** ASP.NET SignalR is a library for real-time web functionality. It enables bidirectional communication between the server and connected clients, making it suitable for building real-time applications.

ASP.NET

1. **Rich Set of Controls:** ASP.NET provides a rich set of server controls that encapsulate common HTML elements. These controls simplify the development process and allow for the creation of dynamic and interactive web pages.
2. **RESTful Services:** ASP.NET Web API enables the creation of RESTful services, allowing developers to build APIs for communication between different parts of an application or with external systems.
3. **Lightweight Web Pages:** Razor Pages is a lightweight page-centric programming model in ASP.NET Core. It simplifies the development of web pages by combining the view and the controller in a single file.
4. **Cross-Platform:** ASP.NET Core is the cross-platform, open-source version of ASP.NET. It supports development on Windows, Linux, and macOS, making it versatile and adaptable to various environments.
5. **Middleware:**ASP.NET Core middleware allows developers to configure components that participate in processing HTTP requests. It provides a flexible and extensible pipeline for handling requests and responses.

ASP.NET

Applications:

1. ASP.NET is used to build scalable, robust, and high-performance Web applications. Listed below are some major uses of ASP.NET:
2. Bundles and minimizes the size of the scripts and style sheets in your application. This feature improves the performance of the Web applications.
3. ASP.NET uses value providers to filter the data. You can also define customized value providers.
4. Provides an excellent support in asynchronous programming; thereby, allowing you to read as well as write HTTP requests and response. Provides support for HTML 5 form types.
5. As ASP.NET is the server-side technology, it executes on the server prior to being sent to the browser.

Comparson and Discussions

Commonalities:

1. **Declarative Nature:**
   Both FP and logical programming are declarative in nature, focusing on expressing what should be done rather than how.
2. **Expressiveness**:
   Both paradigms offer high expressiveness, allowing concise and readable code.
3. **Abstraction**:
   Abstraction is a key concept in both paradigms, facilitating modular and reusable code.

Comparison and Discussions

Functional Programming	Logical Programming
In this programming paradigm, programs are constructed by applying and composing functions.	In this programming paradigm, program statements usually express or represent facts and rules related to problems within a system of formal logic.
These are specially designed to manage and handle symbolic computation and list processing applications.	These are specially designed for fault diagnosis, natural language processing, planning, and machine learning.
Its main aim is to reduce side effects that are accomplished by isolating them from the rest of the software code.	Its main aim is to allow machines to reason because it is very useful for representing knowledge.
It reduces code redundancy, improves modularity, solves complex problems, increases maintainability, etc.	It is data-driven, array-oriented, used to express knowledge, etc.
Testing is much easier as compared to logical programming.	Testing is comparatively more difficult as compared to functional programming.
It simply uses functions.	It simply uses predicates. Here, the predicate is not a function, i.e., it does n have a return value.

Bibliography

References:

1. http://web.cecs.pdx.edu/ black/CS311/ML.html
2. https://www.geeksforgeeks.org/functional-programming-paradigm/
3. https://hackmd.io/@RubAbella/SJA7wBVHB
4. https://www.geeksforgeeks.org/functional-programming-paradigm/
5. https://www.geeksforgeeks.org/difference-between-functional-and-logical-programming/
6. https://dotnet.microsoft.com/en-us/learn/aspnet/what-is-aspnet
7. https://eternitech.com/technologies/asp-net/
8. https://www.linode.com/docs/guides/logic-programming-languages/
9. https://www.youtube.com/watch?v=BfEjDD8mWYg
10. https://chat.openai.com/