CS4022D Principles of Programming Languages
Lecture #6: Programming Paradigms

Saleena N
NIT Calicut

August 9, 2021

History

- 1950s: FORTRAN, COBOL, Algol, LISP
- User Communities: Scientific Computations, Artificial Intelligence, Information Systems
- Influences
  - FORTRAN - efficient translation of algebraic expressions
  - COBOL - records
  - Algol - grammar for formally defining syntax

History

- Difference in the computational problem domains
  - Artificial Intelligence: programs that model human intelligent behavior-Functional Programming, Logic Programming
  - Science and Engineering: Fortran *"Formula Translator"* - efficient and accurate computations
  - Information Systems - need to process large amounts of data-COBOL (Common Business Oriented Language), SQL
  - System Programming - OS, Compilers, Debuggers. real time, embedded systems
  - Educational: Basic, Pascal

Exercise

- Write an algorithm for computing the factorial of a given number $n > 0$

Exercise

- Write an algorithm for computing the factorial of a given number $n > 0$
- Your solution:
  - recursive?
  - itertive?

Programming Paradigm

- Compute factorial - two different solution
  - recursive - functional
  - iterative - procedural
- Underlying thought process that led to the solution - difference in the pattern of thought

Programming Paradigms

- Programming Paradigm: Pattern of problem solving thought
  - Functional
  - Imperative
  - Logic
  - Object-oriented

Imperative Programming

- Imperative - based on the Von Neumann model - stored programs, variables
- program as a series of commands
- assignments, loops, sequences.....
- procedural abstraction

Functional Programming

- Functional - computation viewed as mathematical function mapping input to output

Functional Programming

- Lambda Calculus (Alonzo Church) - foundation of functional programming
- Original functional programming language - LISP developed by John Mc Carthy - theorem proving, rule based systems, earlier AI applications - Scheme is a variant of LISP
- Other languages - ML, Haskell
- Pure Functional Programming - Computation viewed as mathematical function mapping input to output

Functional Programming

```
let rec fact n =
        if (n=0) then 1 else n * fact(n-1)
```

Functional Programming

- Is $f(x) + f(x)$ same as $2 * f(x)$?
- Referential Transparency - A function has referential transparency if its value depends only on the value of its arguments
  - guaranteed in pure functional programming language (no notion of states, no mutable variables, side effect free)
- For Practical reasons, most functional programming languages do support variables, assignment, loops - impure language feature

Imperative Programming

- Notion of program state- collection of (variable, value) pairs - simplified view
- Assignment changes values of variables - causes side effect - change in state
- Referential Transparency can not be guaranteed
  - evaluation of $f(x)$ may update a global variable - a side effect
  - value of $f(x)$ may depend on program state - $f(x) + f(x)$ and $2 * f(x)$ need not result in the same value
- Order of evaluation of parameters in C language function calls

Object-Oriented Programming

- Program as a collection of objects
- objects interact with each other
- objects can change state
- encapsulation, inheritance

Logic Programming

- Logic(declarative) programming - models a problem by declaring what outcome the program should accomplish - rather than how it should be accomplished
- specifications for problem solutions expressed in mathematical logic
- rule-based languages - program's declaration looks like a set of rules
- evolved out of needs in NLP, Automatic Theorem Proving
- Formal foundation- Propositional and Predicate Logic
- Prolog

A Prolog Program

```
speaks(Rohit, Hindi)
speaks(Sandra, Malayalam)
speaks(Ebin, Malayalam)
speaks(Pritish, Hindi)

talkswith(Person1, Person2) :- speaks(Person1, L),
speaks(Person2, L), Person1 \= person2
```

Conclusion

- Some languages support more than one paradigms
- Language choice - problem domain, personal preference, ease of use/implementation, availability of open source compilers and supporting tools....