Principles of Programming Languages

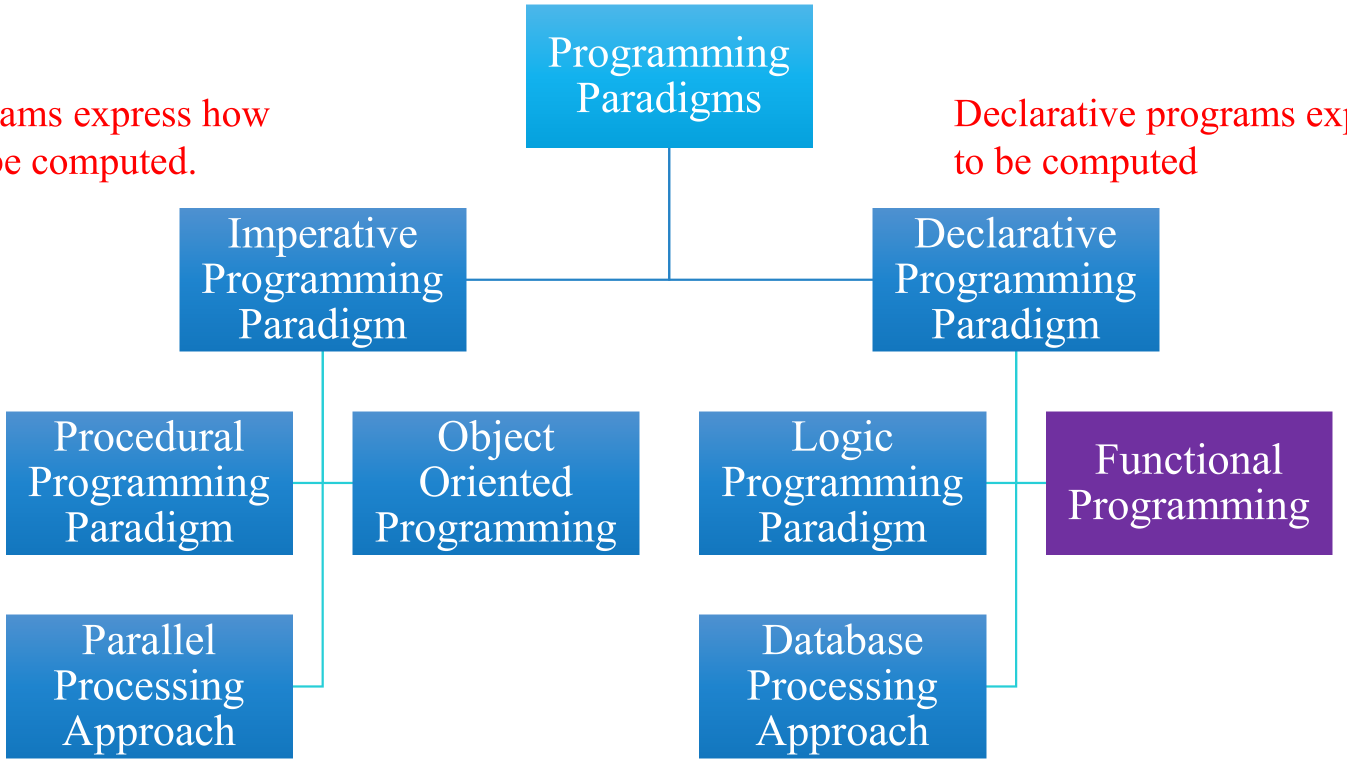
Introduction to Functional Programming Language Paradigm

S6CSE, Department of CSE, Amritapuri

Programming Paradigms

Imperative programs express how something is to be computed.

Declarative programs express what is to be computed


Programming Paradigms
Imperative Programming Paradigm
Declarative Programming Paradigm
Procedural Programming Paradigm
Object Oriented Programming
Logic Programming Paradigm
Functional Programming
Parallel Processing Approach
Database Processing Approach

Functional Programming Paradigm

- Functional programming is a programming paradigm in which we try to bind everything in pure mathematical functions style.
- It is a declarative type of programming style.
- The 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. An expression is evaluated to produce a value whereas a statement is executed to assign variables.

To compare

Java code for summing the integers 1 to 10

```
int total=0;
for (int i=1;1<=10; i++)
        total=total+i;
```

The computation method is variable assignment.

To compare

Summing in 1 to 10 in Haskell

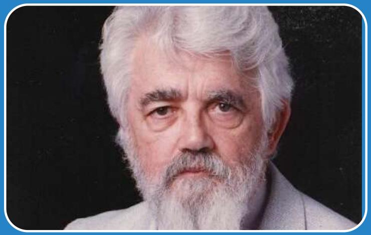
```
sum [1..10]
```

The computation method is functional application.

Historical Background

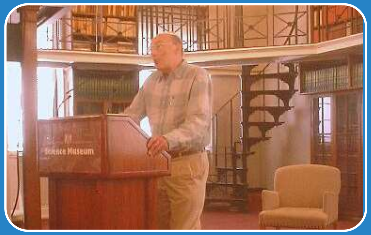
1930s

- Alonzo Church develops the lambda calculus, a simple but powerful theory of functions.



1950s

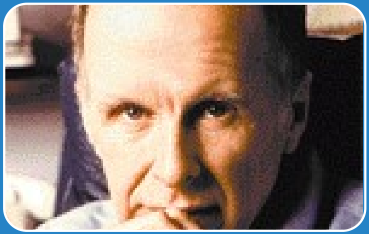
- John McCarthy develops Lisp, the first functional language, with some influences from the lambda calculus, but retaining variable assignments.



1960s

- Peter Landin develops ISWIM, the first pure functional language, based strongly on the lambda calculus, with no assignments.

Historical Background



1970s

- John Backus develops FP, a functional language that emphasizes higher-order functions and reasoning about programs.

1970s

- Robin Milner and others develop ML, the first modern functional language, which introduced type inference and polymorphic types.

1970s - 1980s

- David Turner develops a number of lazy functional languages, culminating in the Miranda system.

Haskell

A Purely Functional Language

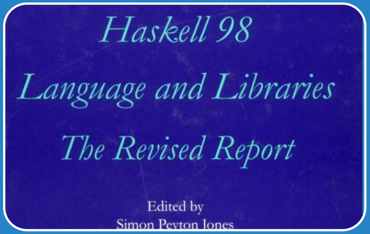
featuring static typing, higher-order functions, polymorphism, type classes and monadic effects

1987: An international committee of researchers initiates the development of Haskell, a standard lazy functional language.

Historical Background

1990s

- Phil Wadler and others develop type classes and monads, two of the main innovations of Haskell.



Haskell 98

Language and Libraries

The Revised Report

Edited by
Simon Peyton Jones

2003

- The committee publishes the Haskell Report, defining a stable version of the language; an updated version was published in 2010.

2010-date

- Standard distribution, library support, new language features, development tools, use in industry, influence on other languages, etc.

Next Lecture – Intro to Haskell