Programming Languages: An Active Learning Approach ©2008, Springer Publishing

Chapter 1 Introduction

Kent D. Lee

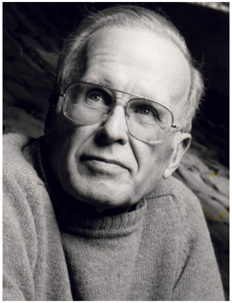
Introduction

- The intent of this text is to introduce you to two new programming paradigms that you probably haven't used before. As you learn to program in these new paradigms you will begin to understand that there are different ways of thinking about problem solving. Each paradigm is useful in some contexts. This book is not meant to be a survey of lots of different languages. Rather, its purpose is to introduce you to the three styles of programming languages.
  - Imperative/Object-Oriented Programming with languages like Java, C++, Ruby, Pascal, Basic, and other languages you probably have used before.
  - Functional Programming with languages like ML, Haskell, Lisp, Scheme, and others.
  - Logic Programming with Prolog.

Historical Perspective

- Sophus Lie found ways of solving Ordinary Differential Equations by exploiting properties of symmetry within the equations.
- While the techniques he discovered were hard for people to learn and use at the time, today computer programs capable of symbolic manipulation use his techniques to solve these and other equally complicated problems.

John Backus



Historical Perspective

- As mathematicians' problem solving techniques became more sophisticated and the problems they were solving became more complex, they were interested in finding automated ways of solving these problems.
- John von Neumann made one of the great contributions to the architecture of modern computers when, in 1944, he wrote a memo suggesting how to encode programs as sequences of numbers resulting in a stored-program computer.
- The architecture is called the von Neumann architecture because of John von Neumann's contributions.
- Around 1958, Algol was created and the second revision of this language, called Algol 60, was the first modern, structured, imperative programming language.

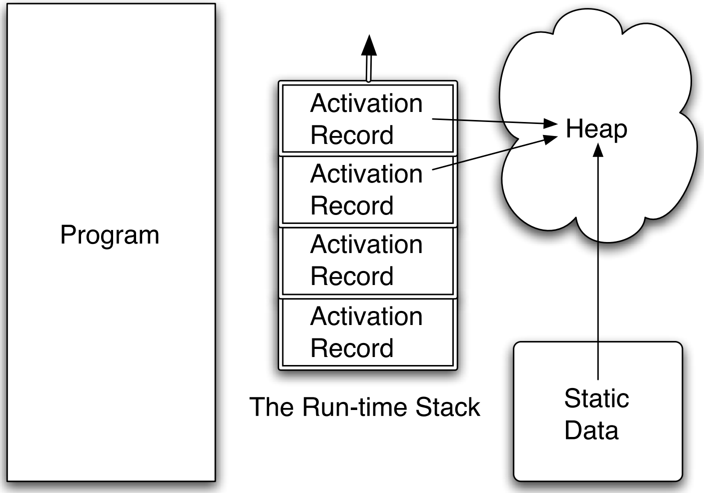
- While the language was designed by a committee, a large part of the success of the project is due to the contributions of John Backus pictured in figure 2. He described the structure of the Algol language using a mathematical notation that would later be called Backus-Naur Format or BNF.

John McCarthy

Historical Perspective

- Alonzo Church was developing a language called the lambda calculus, usually written as the λ-calculus.
- Ideas from the λ-calculus led to the development of Lisp by John McCarthy, pictured in figure 4.
- In contrast to Algol 60, the focus of these languages was on functions and what could be computed using functions.
- Lisp was developed around 1958, the same time that Algol 60 was being developed.
- Many problems were being expressed in terms of propositional logic and first order predicate calculus.
- Prolog is a programming language that grew out of the desire to solve problems using logic.

Conceptual View of the Imperative Model



The Imperative Model

- In the imperative model, memory is divided into regions which hold the program and the data.
- The data area is further subdivided into the static or global data area, the run-time stack, and the heap pictured in figure 1.
- When a program executes it uses a special register called the stack pointer (SP) to point to the top activation record on the run-time stack.
- An activation record contains information about the currently executing function.
- Static or global data sometimes exists and sometimes does not depending on the language.
- The heap is an area for dynamic memory allocation.
- The primary goal of the imperative model is to get data as input, transform it via updates to memory, and then produce output based on this imperatively changed data.

The Imperative Model

☞ Practice 1.1

Find the answers to the following questions.

1. What are the three divisions of data memory called?
2. When does an item in the heap get created?
3. What goes in an activation record?
4. When is an activation record created?
5. When is an activation record deleted?
6. What is the primary goal of imperative, object-oriented programming?

▸ View Solution

The Functional Model

- Functions and parameter passing are the primary means of accomplishing data transformation.
- Data is not changed in the functional model. Instead, new values are constructed from old values.
- The run-time stack becomes more important because most work is accomplished by calling functions.
- Data is certainly dynamically allocated, but once data is created on the heap it cannot be modified in a pure functional model.

The Functional Model

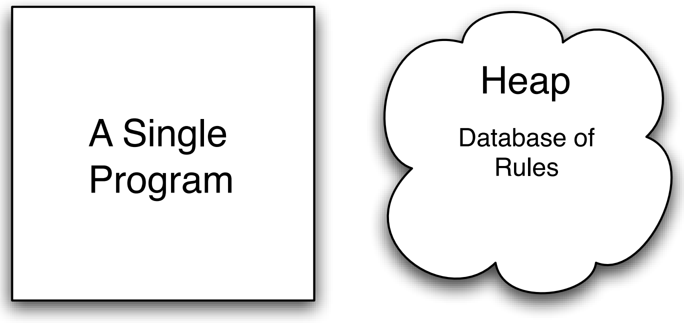
☞ Practice 1.2

Answer the following questions.

1. What are some examples of functional languages?
2. What is the primary difference between the functional and imperative models?
3. Immutable data is data that cannot be changed once created. The presence of immutable data simplifies the conceptual model of programming. Does the imperative or functional model emphasize immutable data?

▸ View Solution

Conceptual View of the Logic Model of Computation



The Logic Model

- In the logic model the programmer doesn’t actually write a program at all. Instead, the programmer provides a database of facts or rules.
- From this database, a single program tries to answer questions with a yes or no answer.
- One can assert new rules and retract rules as the program executes.

The Logic Model

☞ Practice 1.3

Answer these questions on what you just read.

1. How many programs can you write in a logic programming language like Prolog?
2. What does the programmer do when writing in Prolog?

▸ View Solution

Bjarne Stroustrup

A Brief History of C++

- C++ was designed by Bjarne Stroustrup, pictured in figure 1, between 1980 and 1985 while working at Bell Labs.
- While C was efficient, there were other languages that had either been developed or were being developed that encouraged a more structured approach to programming.
- Around 1980, Bjarne Stroustrup began working on the design of C++ as a language that would allow C programmers to keep their old code while allowing new code to be written using these Object-Oriented concepts.
- Today an ANSI committee oversees the continued development of C++ although changes to the standard have slowed in recent years.

A Brief History of Ruby

- Ruby is a relatively new addition as a programming language.
- Yukihiro Matsumoto (Matz for short) created Ruby in 1993 and released it to the public in 1995.
- As Matz was developing Ruby he “wanted a scripting language that was more powerful than Perl, and more object-oriented than Python."
- In Ruby, all data are called objects.

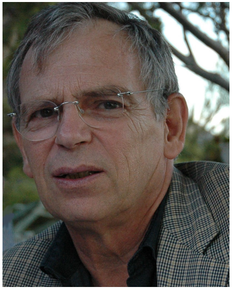
Robin Milner

A Brief History of Standard ML

- Standard ML originated in 1986, but was the follow-on of ML which originated in 1973.
- ML was originally designed for a theorem proving system.
- The theorem prover was called LCF, which stands for Logic for Computable Functions.
- Robin Milner, pictured in figure 2, was the principal designer of the LCF system.
- ML was influenced by Lisp, Algol, and the Pascal programming languages.
- An important facet of ML is the strong type checking provided by the language.
- The type inference system, commonly called Hindley-Milner type inference, statically checks the types of all expressions in the language.

- ML is higher-order supporting functions as first-class values. This means functions may be passed as parameters to functions and returned as values from functions.
- The strong type checking means it is pretty infrequent that you need to debug your code. What a great thing!
- Pattern-matching is used in the specification of functions in ML. Pattern-matching is convenient for writing recursive functions.
- The exception handling system implemented by Standard ML has been proven type safe, meaning that the type system encompasses all possible paths of execution in an ML program.

Alain Colmerauer



A Brief History of Prolog

- Prolog was developed in 1972 by Alain Colmerauer with Philippe Roussel.
- Their research led them to invite Robert Kowalski (pictured in figure 3), who was working in the area of logic programming and had devised an algorithm called SL-Resolution, to work with them in the summer of 1971.
- Colmerauer and Kowalski, while working together in 1971, discovered a way formal grammars could be written as clauses in predicate logic.
- Colmerauer soon devised a way that logic predicates could be used to express grammars that would allow automated theorem provers to parse natural language sentences efficiently.

Robert Kowalski

History of Languages Questions

☞ Practice 1.4

Answer the following questions.

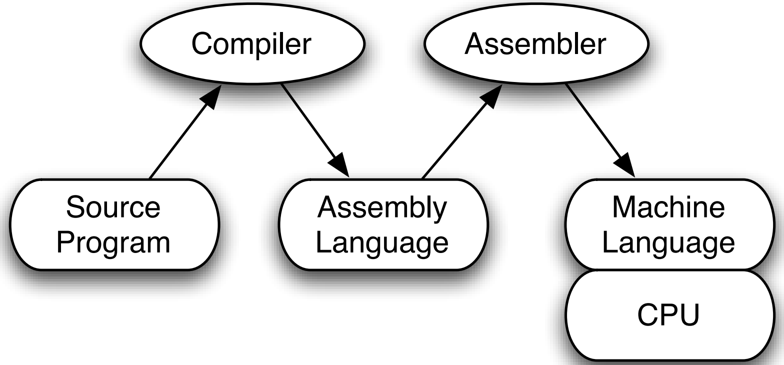
1. Who invented C++? C? Standard ML? Prolog? Ruby?
2. What do Standard ML and Prolog's histories have in common?
3. What language or languages was Ruby based on?

▸ View Solution

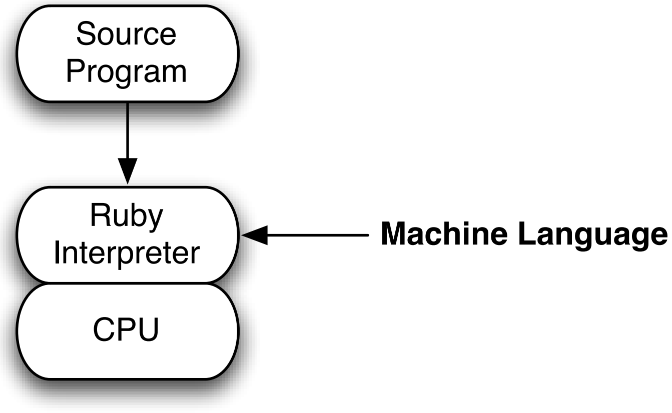
Language Implementation

- Interpreted
- Compiled
- Somewhere in between

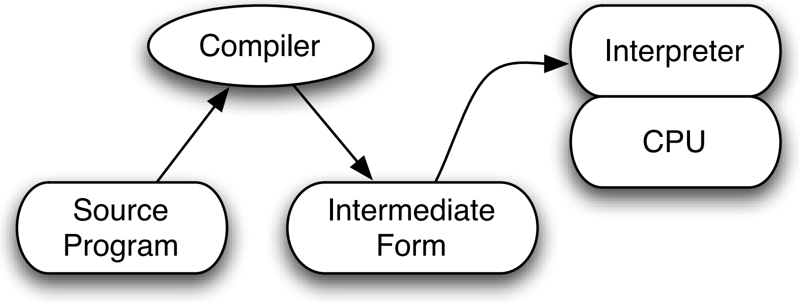
The Compilation Process



The Interpretation Process



Hybrid Language Implementation



Exercises

1. What are the three ways of thinking about programming, often called programming paradigms?
2. Name at least one language for each of the three methods of programming described in the previous question?
3. Name one person who had a great deal to do with the development of the imperative programming model. Name another who contributed to the functional model. Finally, name a person who was responsible for the development of the logic model of programming?
4. What are the primary characteristics of each of the imperative, functional, and logic models?
5. Who are the main people involved in each of the four languages this text covers: C++, Ruby, Standard ML, and Prolog?

6. Where are the people you mentioned in the previous question today? What do they do now?
7. Why is compiling a program preferred over interpreting a program?
8. Why is interpreting a program preferred over compiling a program?
9. What benefits do hybrid languages have over interpreted languages?

Solution to Practice Problem 1.1

1. The run-time stack, global memory, and the heap are the three divisions of data memory.
2. Data on the heap is created at run-time.
3. An activation record holds information like local variables, the program counter, the stack pointer, and other state information necessary for a function invocation.
4. An activation record is created each time a function is called.
5. An activation record is deleted when a function returns.
6. The primary goal of imperative, object-oriented programming is to update memory by updating variables and/or objects as the program executes. The primary operation is memory updates.

Go Back to Practice Problem

Solution to Practice Problem 1.2

1. Functional languages include Standard ML, Lisp, Haskell, and Scheme.
2. In the imperative model the primary operation revolves around updating memory (the assignment statement). In the functional model the primary operation is function application.
3. The functional model emphasizes immutable data. However, some imperative languages have some immutable data as well. For instance, Java strings are immutable.

▸ Go Back to Practice Problem

Solution to Practice Problem 1.3

1. You never write a program in Prolog. You write a database of rules in Prolog that tell the single Prolog program (depth first search) how to proceed.
2. The programmer provides a database of facts and predicates that tell Prolog about a problem. In Prolog the programmer describes the problem instead of programming the solution.

Go Back to Practice Problem

Solution to Practice Problem 1.4

1. C++ was invented by Bjourne Stroustrup. C was created by Dennis Ritchie. Standard ML was primarily designed by Robin Milner. Prolog was designed by Alain Colmerauer and Philippe Roussel with the assistance of Robert Kowalski. Ruby was created by Yukihiro Matsumoto (Matz for short).
2. Standard ML and Prolog were both designed as languages for automated theorem proving first. Then they became general purpose programming languages later.
3. Ruby has many influences, but Smalltalk and Perl stand out as two of the primary influences on the language.

Go Back to Practice Problem