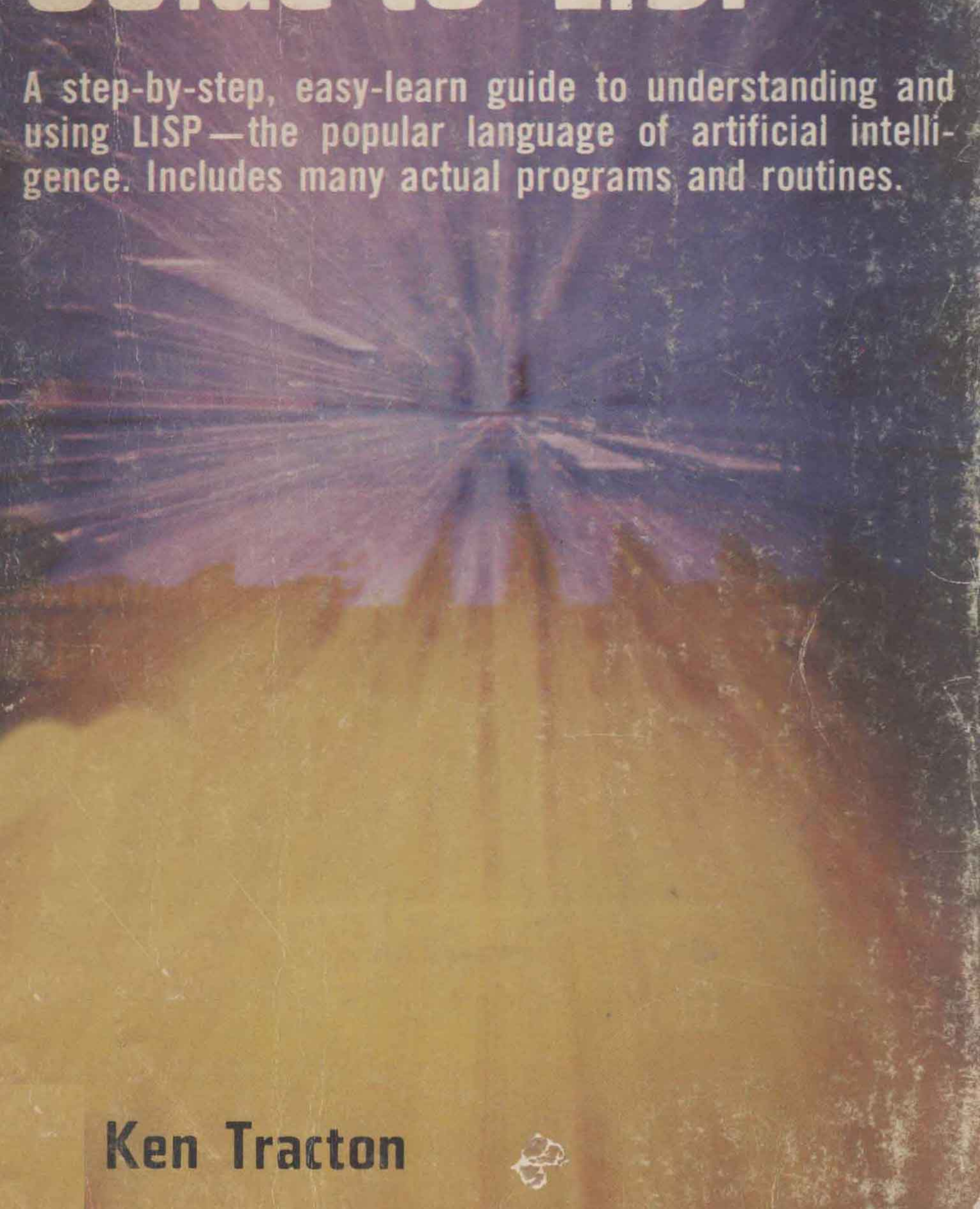
TAB BOOKS/No. 1045 $5.95

Programmer's Guide to LISP

A step-by-step, easy-learn guide to understanding and using LISP—the popular language of artificial intelligence. Includes many actual programs and routines.



Ken Tracton

Programmer's Guide to LISP

DEDICATION

I would like to thank my friends who somehow understood me when I was either wiriting or deep into computer printouts. I would especially like to thank Alec, who coauthors other computer language texts with me.

Programmer's Guide to LISP

by Ken Tracton

TAB BOOKS Inc.
BLUE RIDGE SUMMIT, PA. 17214

FIRST EDITION

FIRST PRINTING—AUGUST 1980

Library of Congress Cataloging in Publication Data

Tracton, Ken.
Programmer's guide to LISP.

Includes index.
1. LISP (Computer program language) I. Title.
QA76.73.L23T7 001.6'424 79-17237
ISBN 0-8306-9761-6
ISBN 0-8306-1045-6 pbk.

Preface

If you have access to a computer with the LISP language, you can start to use LISP as you read this book. The best way to learn any language is to start using it right away.

The first section of this book has a question and answer format. There are also reviews and plenty of examples in this section. The second section covers many diverse programs and routines in LISP.

The reader who sincerely wants to learn and goes through every question and answer will have all the information needed to understand any LISP program.

Near the end of the question and answer sections I state that different LISP systems might handle some things in a slightly different fashion. For example, when reviewing the programs don't be surprised to see in certain places the " ' " symbols. This has exactly the same meaning as the word quote as described in this text. The LISP interpreter used while preparing this book used this symbol instead of the more common quote function. All programs and definitions have been tested on a CDC-CYBER Computer.

Ken Tracton

Contents

Part One: LISP

The LISP language, probably the best-know artificial intelligence language, was invented by John McCarthy. LISP is generally considered to be a process-description language. It is used in artificial intelligence because it is precise, unambiguous and relatively easy to learn. The whole concept of artificial intelligence would be lost if it were not for LISP and the other LISP-like languages.

Statements in LISP (with the exception of what is called the 0-level) are always enclosed in parentheses. Such an enclosure is termed a list and the items contained are called elements of the list. The LISP language is a prefix language where functions preceed their arguments. This is somewhat similar to the RPN notation in calculators but in reverse.

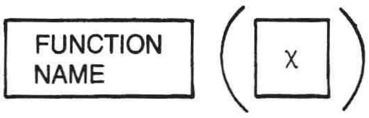
Of the many functions available in LISP, most of them are mnemonics. Care should be taken in learning them in order to avoid confusion. The elements in LISP are generally called ATOMS and lists and ATOMS are termed S-expressions (symbolic-expressions).

There are mathematical functions that use words instead of symbols. Some functions construct lists while other functions take lists apart. There are ways to inhibit function evaluation and ways of causing function evaluation. Execution of the function is termed evaluation. There are also special functions called predicates which are like predicates in languages. These functions are either true or false. Usually the word false is replaced with the word NIL.

Is the logarithm of X written in BASIC as LOG(X)?

Yes, all functions in BASIC are written as "function" name then in parentheses the argument of the function so named (Fig. 1).

Fig. 1.

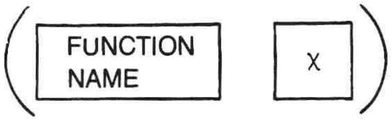


Is the logarithm of X written in LISP as LOG(X)?

No, all functions in LISP are written as ("function name" "variable") (Fig. 2). Therefore the logarithm of X is written as:

(LOG X)

Fig. 2.



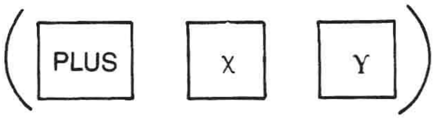
Does this apply to all mathematical functions in LISP?

Yes, all functions are enclosed in parentheses.

How is the sum of X and Y written in LISP (Fig. 3)?

(PLUS X Y)

Fig. 3.



In regards to algebraic manipulation how is LISP different from BASIC?

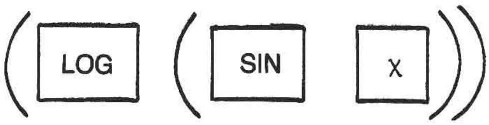
BASIC, FORTRAN and APL use an algebraic language. The common functions such as sum, difference, quotient and product are handled in a special way in order to increase the speed in which the program can be written. In these algebraic languages, primitive algebraic functions are written without parentheses. LISP, however, is a "functional" language. All functions, primitive or not, are written with parentheses.

How do you write the log of sine of X in LISP?

Write with the parentheses (Fig. 4):

(LOG(SIN X))

Fig. 4.



How would you write X/Z-Y in LISP?
Write this expression as:
(QUOTIENT X (DIFFERENCE Z Y))
How would you write X-Y *Z in LISP?
You would write (Fig. 5):
(DIFFERENCE X (TIMES Y Z))

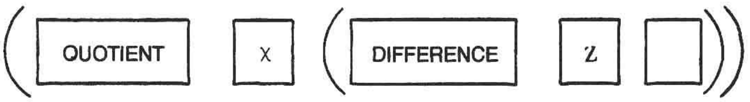


Fig. 5.

Can any arithmetic expression be written in LISP in this way?
Yes, any function that is arithmetic in nature, including parenthesized expressions, can be written in this fashion.
How would you write (A-B) * (Z-X)?
Write this expression as (Fig. 6):
(TIMES(DIFFERENCE A B) (DIFFERENCE Z X))

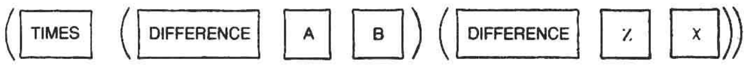
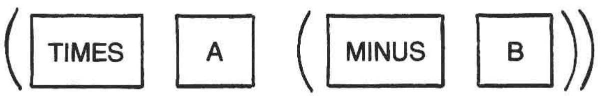


Fig. 6.

Can you write A *(-B) in LISP?
Yes, write A * (-B) as (Fig. 7):
(TIMES A (MINUS B))

Fig. 7.



What is the difference between "DIFFERENCE" and "MINUS"?
The DIFFERENCE function is the same as ordinary subtraction, while the "unary minus" function is written as (Fig. 8):

(MINUS "variable")
Therefore, to denote a negative number in LISP write:
(MINUS X)
And never (-X)

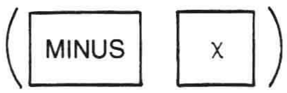


Fig. 8.

Do the algebraic functions in LISP always have two arguments?

Yes, all algebraic functions in LISP must have a minimum of two arguments, except for the unary minus function.

How do you write the function that forms the sum of three or more quantities?

You write (Fig. 9):
(PLUS A B C)
which has the same meaning as:
(PLUS A(PLUS B C))

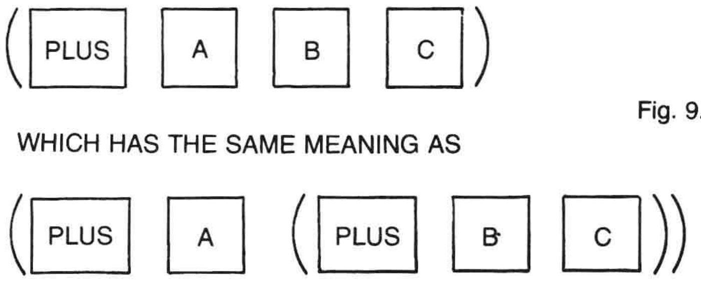


Fig. 9.

Does this procedure also apply to multiplication?

Yes, you can write X * Y * Z as (Fig. 10):
(TIMES X Y Z) instead of writing:
(TIMES X (TIMES Y Z))

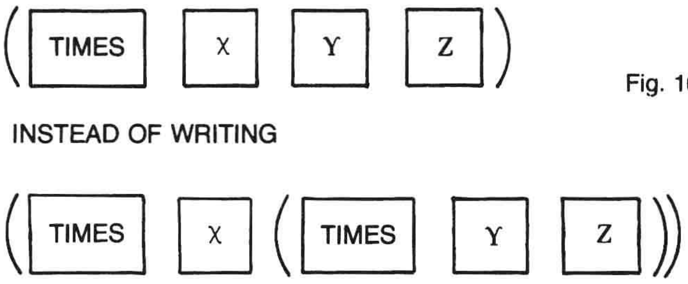


Fig. 10.

What are the arithmetic functions in LISP?

They are PLUS (addition), DIFFERENCE (subtraction), TIMES (multiplication), QUOTIENT (division) and REMAINDER (the value that remains after division).

What exactly is the REMAINDER function?

The REMAINDER function returns as its value the "remainder" when X is divided by Y. An example would be (Fig. 11):

(REMAINDER X Y)

if X and Y are 12 and 5 respectively, the remainder would be 2.

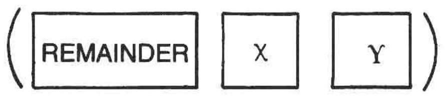


Fig. 11.

Are there any other mathematical functions available in LISP?

Yes, but they are discussed later.

EXPRESSIONS

Does the use of functions in LISP have disadvantages or advantages?

There are both disadvantages and advantages to the use of functions in the LISP manner.

What are the disadvantages?

The main disadvantages of the "function" method of LISP are the lengthier expressions and the need for more parentheses (Fig. 12). For example, it is certainly easier to write in BASIC:

SIN (X * X-4)*A/B

than it is to write in LISP:

(TIMES(SIN(DIFFERENCE(TIMES X X)4)) (QUOTIENT A B))

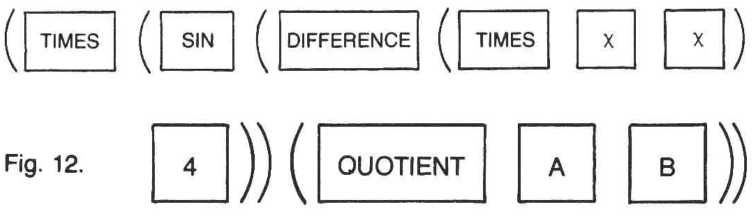


Fig. 12.

What are the advantages of functions?

The advantage of the functional notation in LISP is its ability to unify the language. In LISP, this is the only type of construction available. Every feature of LISP, such as the transferring of control, conditional tests, definitions and so forth are "defined" by creating a special function to handle them.

Is it always necessary to use all those parentheses for simple arithmetic?

Yes, for standard LISP. But there is a special version of LISP, called MLISP, which was designed to implement arithmetic functions without resorting to "all" those parentheses.

What are the expressions called that you have been using so far?

They are termed S-expressions.

How many component types are found in an S-expression?

Two, they are the "elements" such as numerics (1 2 3.....9) and character symbols (A B C . . . . Z). The other component is the parenthesis (Fig. 13).

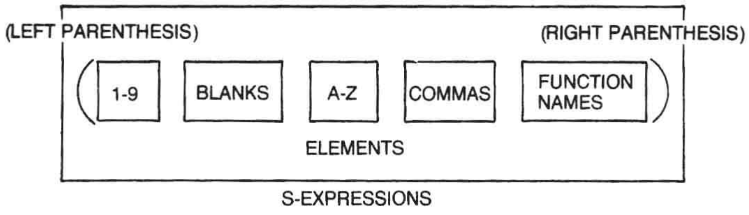


Fig. 13.

What do you call the numerics and symbols in LISP?

They are called ATOMS.

Is every expression that contains ATOMS and parentheses an S-expression?

No, an S-expression only exists if certain rules are followed.

Are the following examples S-expressions?

```
(TIMES V N))
)PLUS 4 5)
(DIFFERENCE A B)
```

The first two are not S-expressions because the first example ends in two parentheses and the second has a reversed first parenthesis. The third expression is indeed an S-expression.

How do you define an S-expression?

An S-expression is made up of ATOMS and parentheses. There must be a balance between the number of left parentheses and the number of right parentheses. There *must* be no reversal of parentheses and the parentheses *must not* be in the wrong spot.

Is there another important aspect of the S-expression?

Yes, you must also consider the blanks. In BASIC and in FORTRAN blanks are ignored. But in LISP blanks are very important to the operation of the functions.

What are blanks used for?

Blanks are used to separate arguments.

Can anything else be used instead of blanks in LISP?

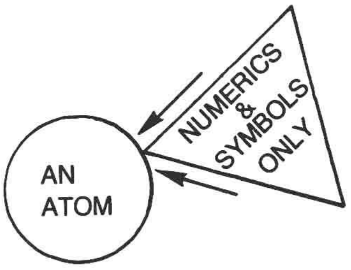
Commas can be used instead, but most programmers prefer to use blanks.

In what ways can commas replace blanks?

A comma can be placed in any spot where a blank would have been (Fig. 14). The following examples will explain:

```
(TIMES C B)..........(TIMES, C B)
                     (TIMES C,B)
                     (TIMES,C,B)
```

Fig. 14.



LIKE CLASSICAL PHYSICS, A LISP ATOM CAN NOT BE DIVIDED TO ANYTHING SIMPLER

Can more than one blank be used in a given place and can blanks ever be left out?

In most programming languages, a string of blanks can be inserted anywhere providing at least one blank is required in the first place. Blanks can also be left out before or after a left or right parenthesis.

Are the following two S-expressions legal in LISP?

(TIMES (SIN (DIFFERENCE (TIMES X X) 4)) (QUOTIENT A B))

(TIMES (PLUS X C) H (PLUS A B) (PLUS R T))

Yes, they are both legal and acceptable in LISP. Only the number of blanks have been increased. The following S-expression is legal because only the blanks before the left and right parentheses have been removed:

(TIMES(PLUS X C) H(PLUS A B)(PLUS R T))

Can the above rules be applied with any function in LISP?

Yes, the rule applies to any construction in LISP.

ATOMS

What are the constants and variables in an S-expression called?

They are called ATOMS

What rules do variables follow in LISP?

Variables follow the standard rule for identifiers. They *must* be composed of letters or numbers and they *must* start with a letter only.

What is the difference between variables in LISP and in most other languages?

The major difference is in the upper amount of characters permitted in the variable name. Most languages only allow a certain amount of characters to be used. In LISP, any amount of characters can be used in the construction of a variable name (Fig. 15).

```
OTHER
LANGUAGES
SHORT VARIABLE NAME
```

Fig. 15.

```
LISP        THIS IS A LONG VARIABLE NAME
```

Are there any special identifiers in LISP?

Yes, T, F and NIL.

What happens if you use one of these special (reserved) identifiers?

Typically, the results will be unpredictable.

What are these reserved variables or identifiers used for?

The T and F mean "true" and "false" respectively, while the NIL has a number of uses.

What are the uses of the NIL in LISP?

Most of the uses of NIL will be covered later, but one is worth mentioning now. Quite often in LISP, NIL is used to replace or substitute for F. Actually, F is not used very often to mean "false". NIL is used instead.