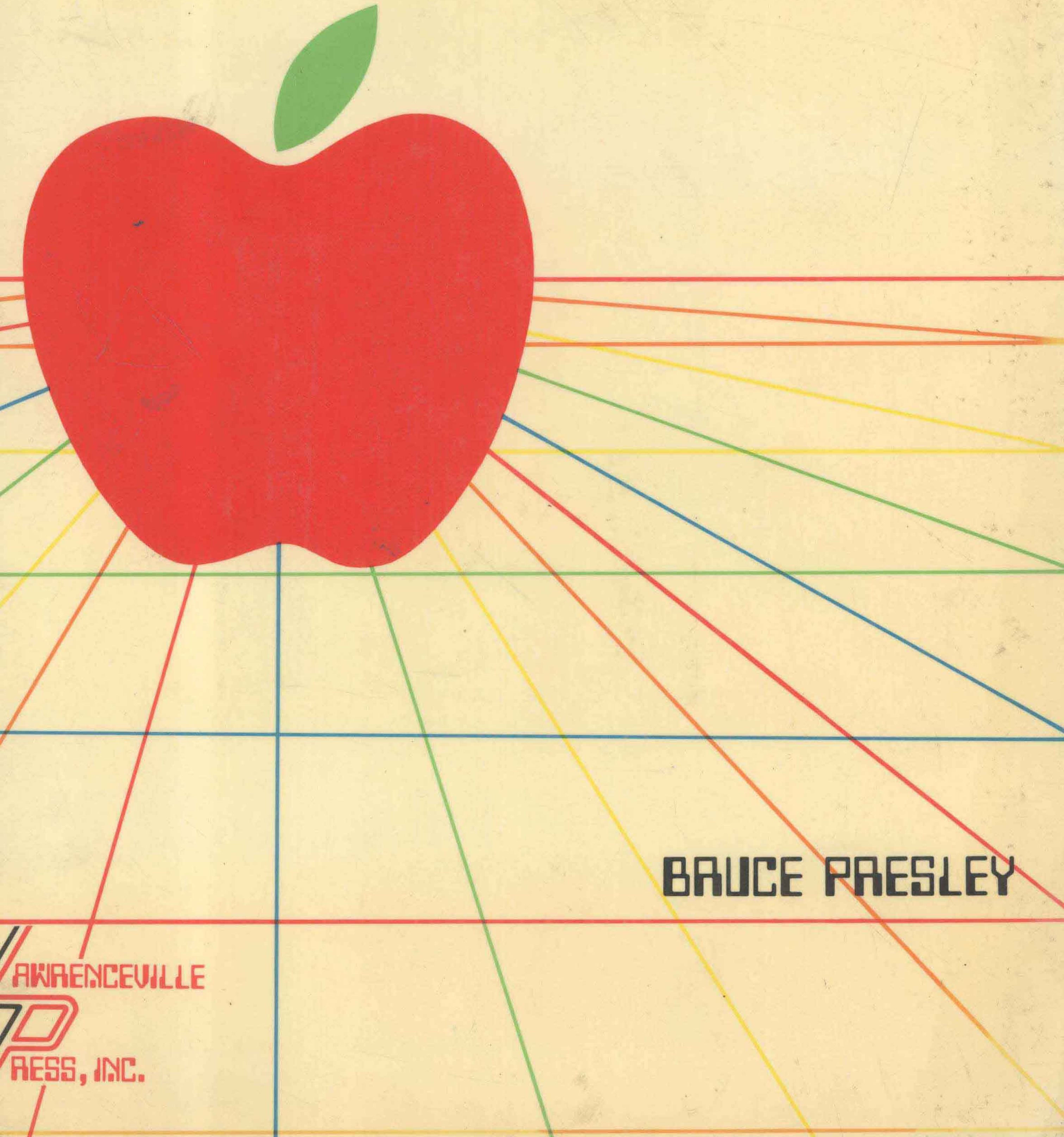
A GUIDE TO PROGRAMMING IN APPLE SOFT®



BRUCE PRESLEY

LAWRENCEVILLE PRESS, INC.

A GUIDE TO PROGRAMMING IN APPLE SOFT

LAWRENCEVILLE PRESS, INC.

BRUCE PRESLEY

DISTRIBUTED BY

VNR VAN NOSTRAND REINHOLD COMPANY

NEW YORK CINCINNATI TORONTO LONDON MELBOURNE

First published in 1982

Copyright ©1982
by

LAWRENCEVILLE
PRESS, INC.

Library of Congress Catalog Card Numer 81-52974

ISBN 0-442-25890-9

VNR

Educational orders may be placed by writing or calling:

Van Nostrand Reinhold Inc.
76 25 Empire Drive
Attn: Department T
Florence, Kentucky 41042
Telephone 606-525-6600

Van Nostrand Reinhold Company
135 West 50th Street, New York, NY 10020

Van Nostrand Reinhold Ltd.
1410 Birchmount Road, Scarborough, Ontario M1P 2E7

Van Nostrand Reinhold Australia Pty. Ltd.
17 Queen Street, Mitcham, Victoria 3132

Van Nostrand Reinhold Company Ltd.
Molly Millars Lane, Wokingham, Berkshire, England RG11 2PY

16 15 14 13 12 11 10 9 8 7 6

PREFACE

The publication of this first edition of A GUIDE TO PROGRAMMING IN APPLESOFT represents the latest achievement of faculty members and students at The Lawrenceville School in developing curricular materials to assist students in learning to program computers.

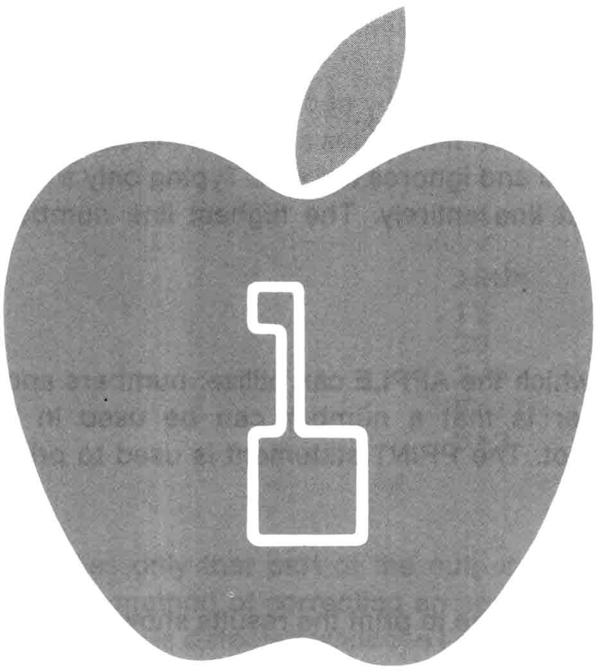
Based on experience gained in our classrooms over the past ten years, we are convinced that a structured, concisely written computer text is invaluable in teaching and learning programming. This latest manual, written specifically to instruct students in Applesoft, not only retains but expands on the features which have resulted in the wide acceptance of our previous publication, A GUIDE TO PROGRAMMING IN BASIC PLUS.

The production of this manual has involved the cooperation and talents of many individuals. Lawrenceville faculty members Stuart Hayes and William Graupner, along with former Lawrenceville students John Partridge, Lester Waters, David Hayden and Robert Lynch, have all made contributions to this publication which have been invaluable. Each has authored and taken responsibility for specific areas of the manual. The clarity and accuracy of each chapter reflect their considerable effort and expertise. Marjorie Vining, of the Rhode Island School of Design, has produced the imaginative cover design and art work, while Dr. Theodore Fraser of The College of the Holy Cross has edited the text to ensure that it is both grammatically and stylistically correct.

To all of these people I am deeply grateful. Thanks to their combined efforts, we have produced what I believe to be one of the very finest computer texts available today.

Bruce W. Presley

TABLE OF CONTENTS



INTRODUCTION TO PROGRAMMING

To communicate with a computer a special language is required. The language of the APPLE computer is called APPLESOFT BASIC which consists of simple English words that the computer understands. In this chapter the user will be introduced to the most fundamental statements required to write a program on the APPLE.

What Is a Program?

A program is a sequence of instructions that informs the computer of the tasks which the user wants it to perform. From the very beginning it is important to realize that the user must determine the order in which the computer performs all tasks assigned to it. It is also essential to realize that the computer operates with a limited vocabulary and that it can understand only certain key words which serve as commands. Hence, words which make perfect sense to the user may make no sense at all to the computer if they are not part of its vocabulary. It is the purpose of this manual to teach one of the special vocabularies of the APPLE and to explain how it can be used in a variety of ways.

Line Numbering

To establish the sequence of instructions for any given program, line numbers must first be assigned to each line of the program. This operation is essential since the computer reads instructions beginning with the lowest line number, then progresses to the next higher number, and ends with the highest numbered line.

It is permissible to type the program lines out of order because the computer puts them back into their proper order. For example, line 20 may have been typed before line 10; but, when the program is run, the computer automatically reads line 10 first. Such a system has the advantage of allowing any forgotten instructions, say between lines 10 and 20, to be entered later as lines 15, or 12, or 17, etc. For this reason it is best to number the lines of the program in units of ten (10, 20, 30, etc.) since this leaves nine possible line numbers that can later be inserted. It is also important to realize that the capital letter O cannot be used in place of zero.

An error in any instruction may be corrected at any time by depressing the return key and retyping the entire line, using the same line number. If two lines are given the same line number, the computer uses only the last one typed and ignores the first. Typing only a line number followed by a RETURN eliminates that line entirely. The highest line number allowed on the APPLE is 63999.

PRINT

There are two basic types of information which the APPLE can utilize: numbers and words. What distinguishes one from the other is that a number can be used in a mathematical calculation, whereas a word cannot. The PRINT statement is used to print both numbers and words.

PROGRAM 1.1

This program uses the PRINT statement at each line to print the results shown below.

```
10   PRINT 5 * 3
20   PRINT 12 + 8
30   PRINT 7 / 2
40   PRINT "JONES"
50   PRINT "5*3"

]RUN
15
20
3.5
JONES
5*3
```

Line 10 instructs the computer to print the result of 5 multiplied by 3. Note that an asterisk (*) is used to indicate multiplication. Line 20 adds 12 and 8 while line 30 divides 7 by 2, with the slash (/) indicating division. At line 40 the word JONES is printed. To be printed, a word or any set of characters must be enclosed within quotation marks. Line 50 prints 5*3 rather than 15 because quotation marks are used. Information which is enclosed in quotation marks is called a string. Both lines 40 and 50, for example, contain strings.

The command RUN is typed after a program has been completed to cause the programmed instructions to be executed. It is not part of the program.

Typing Errors

An error in a program can be corrected in one of two ways. First, the program line can be retyped using the same line number. For example,

```
]40 PRINT "SMITH"

]RUN
15
20
3.5
SMITH
5*3
```

changes only that part of the output of Program 1.1 which is produced by line 40. The second method of correcting an error uses the keys marked → and ← and may be used if the error is detected while the line containing the error is being typed. Pressing the ← key a sufficient number of times will move the cursor on the screen back to the location of the error, where typing the correct symbol will replace the error. Pressing the → key will bring the cursor to the end of the incomplete line so that typing can resume.

LIST

The LIST command is used to print the program currently in the computer's memory. Typing LIST followed by a RETURN will print the entire program, while LIST followed by a line number and a RETURN will cause only that line to be printed. To list portions of a program type LIST followed by the first and last line numbers of the portion desired, separating the two numbers with a hyphen (-). For example:

```
]LIST

10  PRINT 5 * 3
20  PRINT 12 + 8
30  PRINT 7 / 2
40  PRINT "JONES"
50  PRINT "5*3"

]LIST 50

50  PRINT "5*3"

]LIST 20-40

20  PRINT 12 + 8
30  PRINT 7 / 2
40  PRINT "JONES"
```

NEW

Before entering a program into the computer, the NEW command should be used to clear the computer's memory. The use of this command insures that program lines from a previous program do not affect the new program.

```
]LIST

10  PRINT 5 * 3
20  PRINT 12 + 8
30  PRINT 7 / 2
40  PRINT "JONES"
50  PRINT "5*3"

]NEW

]LIST


]
```

Observe that after the NEW command is used, a subsequent LIST shows that no program is currently in the computer's memory.

What Is a Variable?

One of the most useful properties of a computer is its ability to manipulate variables. A variable is a symbol that may assume many different values. The computer uses variables to represent numbers or strings.

Numeric variables are represented by letters or by letters and numbers. A, D, A1, and B1 are all legal names for numeric variables. The APPLE allows the use of long names for variables, but for the sake of simplicity only a single letter or a single letter followed by a single digit will be used in the early chapters of this manual.

Variables can change in value as the name itself implies. For example, suppose that program line

```
20 X = 5
```

is typed. When the program is run and reaches line 20, X will be assigned the value 5. Suppose a later statement such as

```
50 X = 7
```

is typed. Then at line 50 the value of X will change from 5 to 7.

Just as in algebra, the computer can have one numeric variable defined in terms of another. For example, if Y = 4*X, then Y will equal 20 when X = 5, and Y will equal 28 when X = 7.

PROGRAM 1.2

This program assigns values to the variables X and Y, where Y depends upon X.

```
10 X = 12
20 Y = 3 * X + 5
30  PRINT X,Y
40 X = 15
50 Y = 3 * X + 5
60  PRINT X,Y

]RUN
12              41
15              50
```

The value of X is originally 12 at line 10 and then becomes 15 at line 40. Y changes its value at line 50 because X changed at line 40. Note also that a comma used in the PRINT statements at lines 30 and 60 allow both the values of X and Y to be printed on a single line. Note also that up to three variables can be printed on a single line by the use of commas to separate their respective names.

String variable names are used in the same way as numeric variable names, except that a string variable name must end with a dollar sign ($). For example, A$, D$,A1$,C5$ all represent string variables. To assign characters to a string variable the characters must be enclosed in quotation marks. For example,

10 A$ = "HARRY"

At some point later in a program it is possible to assign different characters to the variable. For example,

50 A$ = "SALLY"

PROGRAM 1.3

This program assigns two different sets of characters to the string variable B$.

```
10 B$ = "GEORGE"
20  PRINT B$
30 B$ = "JUDY"
40  PRINT B$
50  PRINT "IT IS NICE TO SEE YOU ";B$

]RUN
GEORGE
JUDY
IT IS NICE TO SEE YOU JUDY
```

Observe that at line 50 a sentence can be formed from two strings, the first of which is not given a variable name. Also, observe how the blank spaces in the first string affect what is printed.

REVIEW

1. Write a program that will print the value of Y when Y = 5X + 7 and X = 5.
2. Write a program that will produce the following output, using a single string variable names for "HARRY" and "JUDY" but not for the other words.

```
]RUN
HELLO HARRY
HOW ARE YOU?
JUDY IS LOOKING FOR YOU.
```

READ, DATA

A variable may be directly assigned its value not only by means of a statement such as X = 5, but also by the use of a combination of READ and DATA statements.

PROGRAM 1.4

This program uses READ and DATA statements to assign values to numeric variables.

```
10  READ X,Y,Z
20  PRINT 2 * X + 3 * Y + 8 * Z
30  DATA  3,2,0.5

]RUN
16
```

The READ statement at line 10 instructs the computer to assign numbers to the variables X, Y, Z. The computer finds these numbers in the DATA statement at line 30 and assigns them in the order listed (X = 3, Y = 2, and Z = 0.5).

In a DATA statement only numbers that are expressed in decimal or scientific form are acceptable. For example, the numbers .0032, -5.78, 1050, 2.9E5, and 4.76 E-12 are acceptable. The E stands for "times ten to the power" (e.g., 5.3 E3 = 5300 and 8.72 E-3 = .00872). However, data such as the arithmetic expressions 15/3, 3*5, and 7+2 are not permitted in DATA statements since, unlike PRINT statements, DATA statements allow for no calculations.

GOTO

Suppose that a student has more than one set of values to substitute for the variables X, Y, Z in Program 1.4. To introduce the extra values, line 30 can be retyped as follows:

```
]30 DATA 3,2,0.5,-7,2.5,1E2

]RUN
16
```

Note that while only the first three numbers in the DATA statement are processed, the remaining numbers are not used. To overcome this problem, line 30 could be retyped each time a new set of data is to be run, but this involves unnecessary labor. By including a GOTO statement between lines 20 and 30, the problem can be more easily solved by establishing a loop which allows line 10 to be used over again.

```
]LIST

10   READ X,Y,Z
20   PRINT 2 * X + 3 * Y + 8 * Z
25   GOTO 10
30   DATA  3,2,0.5,-7,2.5,1E2

]RUN
16
793.5

?OUT OF DATA ERROR IN 10
```

Observe that by typing the command LIST, the computer prints the current version of Program 1.4 including the new line. Running Program 1.4 now processes all of the data presented in line 30. Each time the computer reaches line 25, the GOTO statement causes the computer to return to line 10 where the next set of data is assigned to the variables. On the first pass through the loop X = 3, Y = 2, Z = .5; on the second pass X = −7, Y = 2.5, Z = 1E2. However, on the third attempted pass, no additional data is present for assignment to the variables at line 10, and therefore the error message is printed.

The location of the DATA statement within a program is not important; it can be placed anywhere. When the computer encounters a READ statement it makes use of the DATA statement regardless of its location. The DATA statement may contain strings as well as numbers.

PROGRAM 1.5

This program reads the names and grades of three students and prints the averages of their grades.

```
5  PRINT "NAME","AVERAGE"
10  READ N$,A,B,C,D
20 X = (A + B + C + D) / 4
30  PRINT N$,X
40  GOTO 10
50  DATA  WATERS,83,75,52,80,PARTRIDGE,74,81,92,76,HAYDEN,72,81,83,90

]RUN
NAME            AVERAGE
WATERS          72.5
PARTRIDGE       80.75
HAYDEN          89

?OUT OF DATA ERROR IN 10
```

Examine this program carefully since it contains a number of important concepts. First, note that line 5 produces the headings for the columns. It is placed at the beginning of the program to insure that the headings will only be printed once. Second, note that line 10 assigns a student's name to the string variable N$ and the student's four grades to the numeric variables A, B, C, D. Unless the sequence of string variables and numeric variables in the READ statement is the same as the sequence of strings and numbers in the DATA statement, an error will occur. Third, line 40 returns the program to line 10 to continue to read data. What would happen if at line 40 the statement read GOTO 5 instead of 10?

REVIEW

3. Write a program using READ, DATA statements to evaluate Y where Y = 3X + 5 and X = 3, 5, 12, 17, 8.

4. Write a program containing the DATA line in Program 1.5 which prints only the name and first grade of each student.

```
]RUN
NAME            FIRST GRADE
WATERS          83
PARTRIDGE       74
HAYDEN          72
```

INPUT

In many instances it is preferable to introduce data from the keyboard rather than placing it in a DATA statement. To do this an INPUT statement is used in place of the READ, DATA statements. When an INPUT statement is executed the computer prints a question mark (?) and then waits for data to be entered.

PROGRAM 1.6

Data input from the keyboard is used to assign a value to the variable X in the following program.

```
10   INPUT X
20   PRINT 5 * X * X + 3 * X + 2
30   GOTO 10

]RUN
?3
56
?6
200
?2
28
?

BREAK IN 10
```

The GOTO statement at line 30 creates a loop which will continue to run until it is interrupted by a CONTROL C. To execute CONTROL C, press the CTRL and C keys simultaneously, then press the RETURN key. This halts the run of the program and causes BREAK IN 10 to be printed. If this were not done, Program 1.6 would continue to run until the reset key was pressed or the computer shut off.

PROGRAM 1.7

This program is a revision of Program 1.5. Here a student is asked for his or her name and four grades. The computer then prints the name and grade average. Note that it is possible to have the INPUT statement print a question or a remark by enclosing the words to be printed in quotation marks followed by a semicolon and the variable names.

```
10  INPUT "WHAT IS YOUR NAME? ";N$
20  INPUT "WHAT ARE YOUR FOUR GRADES? ";A,B,C,D
30 X = (A + B + C + D) / 4
40  PRINT N$,X
50  GOTO 10

]RUN
WHAT IS YOUR NAME? SANDRA
WHAT ARE YOUR FOUR GRADES? 78,82,69,75
SANDRA          76
WHAT IS YOUR NAME? JAKE
WHAT ARE YOUR FOUR GRADES? 80,65,79,77
JAKE            75.25
WHAT IS YOUR NAME? TOM
WHAT ARE YOUR FOUR GRADES? 67,53,72,68
TOM             65
WHAT IS YOUR NAME?

BREAK IN 10
```

PRINT Formatting

There are a number of ways in which output can be formatted by the use of punctuation. A few of the more important uses of punctuation are demonstrated by the following program.

PROGRAM 1.8

When using most video displays the APPLE is capable of printing 40 characters on a single line. The number will differ with many paper printers. When the print line consists of 40 characters, this line is divided into three print zones: the first and second are 16 characters wide, the third 8 characters wide. When commas are used in a PRINT statement the output is printed in successive zones.

```
5  PRINT "NAME","GRADES","AVERAGE"
10  READ N$,A,B,C,D
20 X = (A + B + C + D) / 4
30  PRINT
40  PRINT N$,A;" ";B;" ";C;" ";D;"     ";X
50  GOTO 10
60  DATA  WATERS,83,75,52,80,PARTRIDGE,74,81,92,76,HAYDEN,72,81,63,60
```