高等学校试用

英语理工科教材选

第八分册　算法语言

顾成华　杜玉桥　张正楠　李斯和　选编

戴鸣钟　李国瑞　马泰来　主审

机械工业部部属院校选编

高等学校试用

英语理工科教材选

Book Ⅷ

Algorithm

顾成华　杜玉桥　张正楠　李斯和　选编

戴鸣钟　李国瑞　马泰来　主审

机械工业部部属院校选编

本分册内容选自M. BOILLOT·W. HORN所著BASIC英文原版书(1976年版)中的第3、4、5、6四章及Allen B. Tucker, Jr. 所著Programming Languages英文原版书(1977年版)中的第3章。

高等学校试用

英语理工科教材选

(第八分册 算法语言)

顾成华 杜玉桥 张正楠 李斯和 选编

戴鸣钟 李国瑞 马泰来 主审

上海机械学院教务处 出版发行

上海机械学院印刷厂 印刷

成本费 ¥0.71元

编者的话

为了提高机械工业部部属院校学生的外语水平，培养学生阅读英语科技书刊的能力，我们选编了这套“英语理工科教材选”。整套“教材选”共分九个分册，内容包括数学、物理、理论力学、材料力学(与理论力学合为一个分册)、电工学、工业电子学、金属工艺学、机械原理、机械另件(与机械原理合为一个分册)、计算机算法语言、管理工程等十一门业务课程。各业务课都选了三章英语原版教材（个别也有选四章），供机械工业部部属院校试用。

在业务课中使用部分外语原版教材，这是我们的一次尝试，也是业务课教材改革、汲收国外先进科学技术的探索。在选材时，我们考虑了我国现行各课程的体系、内容以及学生的外语程度，尽可能选用适合我国实际的外国材料。

本“教材选”的选编工作，是在机械工业部教育局的直接领导下，由部属院校的有关教研室做了大量调查研究后选定的，并进行注释和词汇整理工作。由马泰来、卢思源、李国瑞、柯秉衡、谢卓杰、戴炜华、戴鸣钟等同志(以姓氏笔划为序)组成的审编小组，对选材的文字、注释、词汇作了审校。戴鸣钟教授担任整套“教材选”的总审。

由于时间仓促，选材、注释和编辑必有不尽完善之处，希广大读者提出宝贵意见，以利改进。

1983年4月

CONTENTS

3 BASIC STATEMENTS: LET, PRINT, END, TAB

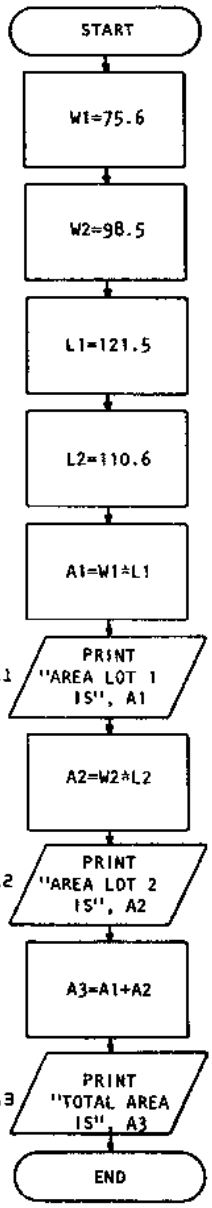
3—1 PROBLEM EXAMPLE

Consider the following problem:

Mr. X has two rectangular lots of land. The width of lot 1 (W1) is 75.6, and its length (L1) is 121.5. The width of lot 2 (W2) is 98.5, and its length (L2) is 110.6. Calculate and print the area of each lot and the combined area of both lots. An example of a BASIC program to solve the above problem is shown in Figure 3-1.

Note the three types of BASIC statements in the BASIC program shown in Figure 3-1:

1. The replacement statement with key word LET,
2. The output statement with key word PRINT, and
3. The termination statement with key word END.




```
10 LET W1=75.6
20 LET W2=98.5
30 LET L1=121.5
40 LET L2=110.6
50 LET A1=W1*L1
60 PRINT "AREA LOT 1 IS",A1
70 LET A2=W2*L2
80 PRINT "AREA LOT 2 IS",A2
90 LET A3=A1+A2
95 PRINT "TOTAL AREA IS",A3
97 END
RUN
```

Set the variables to the respective lengths and widths of both lots. Each statement (line of code) should be preceded by a line number and concluded by a carriage return, that is, press the RETURN KEY. See Figure 3-2.

Compute area of first lot A1.

Print out message "AREA LOT 1 IS", followed by A1, which is the result.

Compute area of second lot A2.

Print out message "AREA LOT 2 IS", followed by A2, which is the result.

Compute combined area A3.

Print out message "TOTAL AREA IS", followed by A3, which is the result.

The RUN instruction differs from the preceding BASIC statements in that it is a system command. It tells the computer to process or execute the program. Note that RUN has no numeric label. A carriage return (RETURN) must always follow the RUN command.

```
AREA LOT 1 IS    9185.4
AREA LOT 2 IS   10894.1
TOTAL AREA IS   20079.5
```

The results are then printed on the teletype below the RUN command.

FIGURE 3—1 Calculation of the Area of Two Lots

Note also the system command RUN which is an instruction addressed to the BASIC operating system. It causes the BASIC program to be executed.

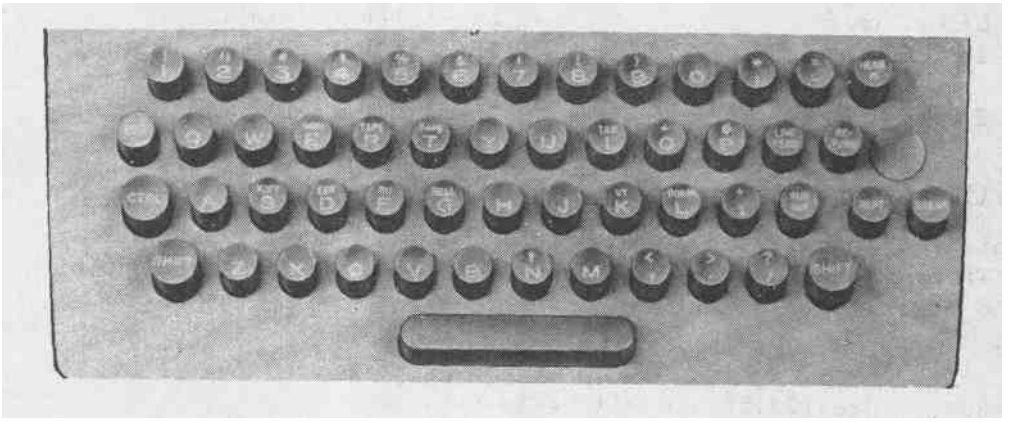


FIGURE 3—2 Teletype Keyboard

Courtesy of Teletype Corporation.

3—2 BASIC STATEMENTS

3—2—1 Replacement Statement LET

As defined in Section 2-2-3, a replacement statement specifies the arithmetic operations to be performed on constants and/or variables and the location (variable) into which the value computed is to be placed. The general form of a BASIC replacement statement is

```
statement-number LET variable=expression
```

where LET is a BASIC keyword.

(In some versions of BASIC the LET keyword is optional.) The value of the expression is computed and placed in the variable in the left-hand part of the statement.

Examples of valid replacement statements:

`5 LET X=3.123`	Define X to be 3.123. (Place 3.123 in location X.)
`10 LET C1=(A+B)/C`	Compute A + B, divide by C, and call result C1.
`20 LET Z=3 ↑ 2`	3 ↑ 2 means 3^2, let Z be equal to 3 squared. (See footnote 1.)
`40 LET K=SQR(Z+4)`	K is computed as the square root of Z + 4.

[1] Some BASIC systems use ** (two multiplication signs) instead of ↑.

Examples of invalid replacement statements:

`10 X=3`	The statement LET is missing.
`LET Y=3/2`	No statement-number.
`15 LET X+Y=1`	Invalid variable on the left-hand side of = sign.
`20 LET 1=X`	Invalid variable on the left-hand side of = sign.

3—2—1a Constants

A constant is a quantity whose numerical value is fixed and explicitly stated. These values can be expressed with or without decimal points and may be positive or negative.

Examples:

-300, .0056,123.56 are all valid constants.

3—2—1b Variables

A variable is the name given to a quantity which can assume different numerical values. Variable names are limited to a single letter or a letter followed by a single digit.[2]

Examples:

X, Z, W1, A0 (0 = zero) are valid variable names.
1A is invalid; it does not start with a letter.
PAY is invalid; it has too many characters.
A¢ is invalid; ¢ is a special character and not a digit.

3—2—1c Expressions

An expression is any combination of constants and/or variables linked by arithmetic operators. Parentheses may be included to denote the order of computations. As stated in Chapter 1, the arithmetic operators are +, -, *, /, and ↑ or ** for exponentiation.

Examples of valid expressions:

Basic Expression	*Corresponding Algebraic Expression*
`(A/B)*C`	$\frac{a}{b} \cdot c$
`A*B-C`	$a \cdot b - c$

[2] Some BASIC systems allow the characters $, #, and @ as legal characters, that is, $3 and #5 are valid variable names.

`-C`	$-c$
`(A*B)↑2`	$(a \cdot b)^2$
`(-C+B)*D`	$(-c + b)d$
`A**B`	a^b

Examples of invalid expressions:

`3(A+B)`	Operator missing; should be 3*(A+B).
`A-(B+C*(K)`	Unpaired parentheses; should be A-(B+C*(K)).
`IN-4.13`	Invalid variable name.
`X*-3`	Two operators side by side; should be X*(-3).

3—2—1d Statement Numbers

Each BASIC statement must be numbered. Statement numbers must be positive integers not exceeding five digits. In some smaller systems statements must be entered in ascending order. In most systems statements may be entered in any desired order; the system will sort the statements in order when the program is listed or executed. Statements will be processed by the computer in ascending order unless a decision statement (IF) or unconditional transfer (GO TO) is encountered. These statements are discussed in Chapter 4.

Example: If you typed in the following statements in the order indicated

```
10 LET X=3
20 PRINT X,Y
15 LET Y=X↑3
```

BASIC would rearrange these statements in the following order:

```
10 LET X=3
15 LET Y=X↑3
20 PRINT X,Y
```

3—2—2 Evaluation of Expressions

As we have already seen, an expression consists of variables and constants linked together by the arithmetic operators +, -, *, /, and ↑ or ** for exponentiation. Since the computer cannot perform more than one operation at the same time, operations are carried out along certain

priorities. When no parentheses are present in an expression, operations are performed from left to right in the following order: exponentiation, then multiplication/division, then addition/subtraction.

Examples:

1. `A+B*C` — Since multiplication has priority, B*C is computed. The result is then added to A giving A+(B*C).

   `3+2*3` = 3+(2*3) = 3+6 = 9.

2. `A*B/C` — Since the two operations have identical priority, the expression is evaluated left to right taking one operation at a time, that is, (A*B) and then (A*B)/C.

   `3*2/3` = $\frac{(3*2)}{3} = \frac{6}{3} = 2$.

3. `(A+B)/C*D` — The parentheses indicate that the sum of A and B is to be performed first. The sum is then divided by C, and the result is multiplied by D giving $\frac{(A*B)}{C}*D$ and not $\frac{(A+B)}{C*D}$.

   `(3+6)/3*6` = $\frac{9}{3}*6 = 18$.

4. `A/B/C` — First, A/B is performed, and the result is then divided by C, which is the same as A/(B*C).

   `8/4/2` = $\frac{8}{4} \div 2 = 2 \div 2 = 1$.

5. `A↑2*C` — First compute A^2, then multiply by C, that is, ↑ has priority over * in the same sense as the * in A*2+C has priority over the +.

   `3↑2*4` $3^2*4 = 36$.

3—2—3 Output Statement PRINT

The general form of an output statement is

statement-number `PRINT` expression-list

where PRINT is a BASIC keyword.

3—2—3a The Expression-List

An expression-list in an output statement can either be

1. A variable,
2. An expression,
3. A string literal (string literals must be enclosed in quotes), or a
4. Combination of any of the above three items separated by commas or semicolons.

Examples of valid PRINT statements:

```
10 PRINT X                      The value for X is printed.
15 PRINT "PAYROLL ACCT3"        The string literal PAYROLL
                                 ACCT3 is printed.
16 PRINT (X-3)*Z                (X-3)*Z is computed, then printed
25 PRINT "THE VALUE OF X IS",X↑3
                                The message or string literal
                                 THE VALUE OF X IS, followed by
                                 the numerical value for X↑3 is
                                 printed.
```

Examples of invalid PRINT statements:

```
15 PRINT X"Y"Z                  No commas separate the variables.
17 PRINT "THE VALUE OF X IS,I3
                                No final quote for literal string.
```

An output line generated by the PRINT statement is divided into zones (see Section 3-5-3). A line, for example, may be divided into 5 zones for a 75-character teletype. (Each zone then allows for 15 spaces for each output item in the expression-list.) A comma between items in the expression-list means space to the next zone (see Figure 3-3). A semicolon suppresses this spacing (see Section 3-5-3).

3—2—3b Examples of PRINT Statements

Example 1: The PRINT statement in the program of Figure 3-3 uses an expression-list separated by commas.

```
00100 LET Y=3
00110 LET X=501.5               Note the four zones on the
00120 PRINT "X=",X,"Y=",Y        output for the four variables
00130 END                        in the PRINT statement.

RUN

 X=           501.5            Y=            3
Zone 1        Zone 2          Zone 3        Zone 4
```

FIGURE 3—3 Sample Use of Commas in an Expression-List

Example 2: The program of Figure 3-4 illustrates the generation of labels for output values.

```
00100 LET R=5
00110 LET H=45
00120 PRINT "HOURS", "RATE", "PAY"
00130 PRINT H,R,R*H
00140 END

RUN

HOURS          RATE          PAY
 45             5            225
```

FIGURE 3—4 Sample Labeling of Output

3—2—4 Termination Statement END

The general form of the termination statement is

statement-number **END** where END is a BASIC keyword.

This statement causes termination of the user's program. It must be the highest numbered statement in the program (see Figure 3-4).

Example:

```
200 END
```

3—3 YOU MIGHT WANT TO KNOW

1. Does it matter where I start typing my BASIC statements?
   Answer: No, it does not matter. Imbedded blanks are also allowed in statements.
   Example: The following statements are equivalent:

```
10 LET      X      =      C1      +4
10LETX=C1+4
```

2. When I have typed a BASIC statement, how do I go to a new line?
   Answer: Press the RETURN key (see Figure 3-2).

3. I make an error in typing a statement and I want to correct it. What do I do?
   Answer: Errors can be corrected in three different ways:
   a. Erase the last character(s) typed. The method for doing this depends on the particular system being used.

b. Erase the entire line. Generally this can be done by pressing the ESCape key, but this too depends on the system in use. Refer to the operating manual for your system.

c. Press RETURN key and ignore any response or error messages from the system. Type the corrected BASIC statement including the same statement number. This new statement will replace the old statement.
Example:

```
If you want to type          10 LET X=(3+Z/4)↑2
 and you typed               10 LET X=(3-Z/4)↑2

Action: Hit RETURN
 and retype                  10 LET X=(3+Z/4)↑2
```

4. I have typed a BASIC statement and hit the RETURN key. The system stops and on a new line prints ERR XX.
Action: An error has been made in the preceding statement. Consult the BASIC reference manual to determine the error type (XX). Correct the statement and retype entire BASIC statement, then press RETURN.

5. My complete BASIC program has been typed and is executing, that is, RUN has been typed. The computer stops and prints
ERR XX AT YYYY
Meaning: This is an execution or run time error at statement number YYYY. Consult reference manual to determine the nature of the error (XX) and change the logic of the program appropriately.
Example:

```
10 LET X=0
15 LET Y=3/X
20 PRINT Y

RUN

ERR 16 AT 0015          The error code and/or message
                         may differ from system to system.
```

Meaning: 16 is an error code to denote an arithmetic error (division by 0). 15 is the statement at which the error has been made.

Action: [3]At this point the user may replace statement 10 by a new statement by typing

```
10 LET X=10          The user presses RETURN, and
                      types RUN.
RUN

 3                   .3 would be printed.
```

6. Suppose I have corrected all errors in the program which is now working well. I want to obtain a clean listing of the entire program without the statements in error and the corrections.
   Action: Most systems will allow the user to list his program by typing the command LIST followed by pressing the RETURN key.

7. The other day I typed my program and listed it. Curiously enough statements that were not mine made their appearance in my listing and caused run time errors. How can I avoid this situation?
   Answer: To clear memory of any leftover code, type the command NEW. To start a new program, it is always recommended to first type NEW.

8. Suppose I want to obtain a paper tape copy of the program.
   Action: Many systems with slow speed paper tape punchers will allow the user to generate a paper tape copy by typing the command LIST, then depressing the punch ON key on the paper tape punch unit, and finally hitting the RETURN key. To read paper tape into memory, insert paper tape in reader and press the START switch. For other systems, consult the BASIC operational procedures. Many systems will allow storing of the program into special files which may be recalled by the user for subsequent use (see Chapter 11).

9. If I forget to initialize a variable to a particular value, and I use that variable in a calculation what will BASIC do?
   Answer: Most BASIC systems will initialize variables to 0.

10. I can't fit the entire expression-list of the PRINT statement on one line because there is not enough room on that line for the list of variables.
    Action: Break down the PRINT statement into two or more consecutive separately numbered PRINT statements.

11. Can BASIC renumber or resequence the statements of my program?
    Answer: Yes. Most systems will use one of the following commands; commas are sometimes required.

[3] In some systems the command DELETE XX (XX is the statement-number) should first be used before the corrected statement XX can be retyped.

RENUM n1 n2 RESEQUENCE n1 n2 RENUMBER n1 STEP n2	In all cases this means: Renumber the first statement n1 with increments of n2 for the following statements. If n2 is omitted, an increment of 10 is automatically provided.

Example: To renumber an entire program in such a way that the first BASIC statement is numbered 5 and each following statement is incremented by 10, the following command would be used:

```
RENUM 5 10
```

All references to old statement numbers in the program are updated to reflect the new numbering system.

12. Is it permissible to perform exponentiation on negative numbers?
Answer: One should exercise care in writing expressions such as (-4)↑2 since exponentiation may involve logarithmic computations where the log of negative numbers is not defined. Instead use (-4)*(-4) or if the expression is too lengthy, use (ABS(-4))↑2. (ABS means absolute value.) Note that negative exponents are perfectly valid; 4↑(-2) has value 1/16.

3—4 PROGRAMMING EXAMPLES

3—4—1 Finite and Infinite Sums

Consider the sum of the following 1,000 terms

$$\text{SUM} = \frac{1}{2^0} + \frac{1}{2^1} + \frac{1}{2^2} + \frac{1}{2^3} + \cdots + \frac{1}{2^{999}} = 1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots + \frac{1}{2^{999}}$$

This is a geometric progression, and its sum can be computed as follows:

$$\text{SUM} = \frac{a(1 - r^n)}{1 - r}$$

where: a is the first term of the progression ($a = 1$)
r is the ratio of any two consecutive terms ($r = 1/2$)
n is the number of terms to be added ($n = 1{,}000$)

The infinite sum

$1 + \frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \cdots + \frac{1}{2^{999}} + \cdots$ and so forth can be