BASIC PROGRAMMING

Second Edition

By John G. Kemeny
and Thomas E. Kurtz

BASIC Programming

SECOND EDITION

JOHN G. KEMENY
Albert Bradley Third Century Professor
Dartmouth College

THOMAS E. KURTZ
Director, Computation Center
Professor of Mathematics
Dartmouth College

JOHN WILEY & SONS, INC., New York · London · Sydney · Toronto

Library of Congress Catalogue Card Number: 70-138911

ISBN 0-471-46830-4

Printed in the United States of America

18 17 16 15 14 13

BASIC Programming

Preface

Basic Programming is a comprehensive introduction to the art of computer programming. It will be useful as a text in secondary schools, in colleges, and in professional training programs, as well as for self-study. Programming techniques are illustrated through a wide variety of applications, many of them of a "nonmathematical" nature.

Programming must be carried out in a computer language, and we have chosen BASIC. This language is rapidly gaining acceptance as a general-purpose computer language, particularly in modern time-sharing systems. The language is so simple and natural that the reader can spend a minimum of time in mastering it and can concentrate on learning programming techniques.

Most texts on programming fall into one of two categories: (1) a discussion of a particular programming language with only a few contrived examples of its application; and (2) a discussion of a narrow area of application, which is of interest to specialists. In *Basic Programming*, the reader is introduced to the language at an elementary level and can then study applications to a variety of problems, many of them being of significant practical interest.

The core of the text consists of Chapters 0 to 7, which have been extensively revised in this second edition. After a brief general introduction in Chapter 0, the reader is shown two complete programs in Chapter 1. These introduce him to the basic ideas of programming by using commonplace examples, a technique we believe to be superior for the majority of students.

Chapter 2 discusses the relation of BASIC to time-sharing, where the user can actively interact with his program. Notice that the use of BASIC does not require a time-sharing system, but the full power of the language will not be realized without such a discussion. (Only a few examples in the text require on-line interaction between the user and his program—the examples in Chapter 11 are the most important exceptions.) Chapters 3 to 6 continue to introduce further statements of the language BASIC and discuss the most common elementary programming techniques, again, by example. Chapter 6 (a new chapter in this second edition) introduces the student to strings. Strings of alphabetic characters are as important as numbers in today's computer applications.

Chapter 7 discusses some ideas on debugging or finding errors. The use of test cases, tracing, and the provision of adequate documentation is stressed.

Part II consists of applications to more than ten areas widely different in nature. The chapters in this part (except for Chapters 13 and 14) are independent of each other, so that the instructor or reader may select the applications that are of greatest interest to him. Chapters 8 to 12 and Chapter 18 may be mastered with a background of three years of high school mathematics. Chapter 8 is specifically designed to illustrate the usefulness of computers in the secondary mathematics curriculum. Chapter 9 consists of examples from elementary number theory—a topic of equal fascination to the expert and the novice. Chapters 10 and 11 (simulation and games) are of particular interest. They illustrate powerful uses of computing that do not depend on a high level of mathematical competence. Chapter 12 consists of applications to business problems. For example, this chapter contains a single program that can solve a wide variety of problems that are normally dealt with in a mathematics of finance course. Chapter 18 (the last chapter) discusses several topics in order to acquaint the reader with some unorthodox uses of computers, including the use of a simple plotting device.

Chapters 15 to 17 consider three mathematical areas that are normally taught at the college level. The

chapter on calculus (Chapter 17) has proved useful in helping to bridge the gap between theory and application in the most universally important branch of college mathematics. Chapter 15 illustrates the uses of computer programs for the solution of problems in elementary statistics—an area in which computers, today, are indispensable. Chapter 16 considers the applications of vectors and matrices—a field of ever-increasing importance in the social sciences. Particularly in this area, the power and convenience of BASIC (as compared with other computer languages) becomes apparent.

Two new chapters (in this second edition) introduce the student to two extremely important areas of modern computing—files and text processing. Chapter 13 discusses files and how to use them in BASIC and presents several applications to business and information processing. Chapter 14 discusses several applications of text processing, for instance, constructing an index and translating a coded message. Chapter 14 requires a knowledge of Chapter 13.

Each chapter concludes with programming exercises, and the chapters in Part II also contain suggestions for major projects.

We express our deep appreciation to the secretaries of the Mathematics Department at Dartmouth for the preparation of the manuscript, to Ronald Fagin, Kinley Larntz, and Jennifer Kemeny for their valuable proofreading assistance, to William Cogswell for testing a preliminary version of the text in a course, and to Sidney Marshall for his musical assistance.

For assistance in preparing the second edition, we also thank James Willis. Finally, we thank the Dartmouth time-sharing computing system for preparing our index.

John G. Kemeny
Thomas E. Kurtz

Contents

BASIC Programming

0 Introduction

0.1 What is a Computer?

In simplest terms, a *computer* is a "machine" that carries out calculations and logical operations upon command. With this definition even a person operating a desk calculator can be considered a computer. But we usually apply the term to those large and complex configurations of electronic and electromechanical machinery, sometimes costing several millions of dollars, or to pygmy versions of the same thing, that carry out numerical and logical calculations at rates of from many thousands to several millions per second.

Actually, a computer, be it small or very complex, must be able to perform two other functions in addition to that of calculation: it must be able to contain or store the data on which it is performing the calculations, and it must be able to communicate with the external world, first to receive the data of the problem and later to yield the answers or results. The former function is performed by "memories" consisting primarily of *magnetic core storage* and usually supplemented by more capacious devices such as *magnetic drums, disks,* and *tapes,* and more recently by photographic or laser devices that have vast capacities. The latter function is often referred to as "input-output" and is carried out by such devices as *card readers* and *punches, magnetic tapes, printers, paper tape* mechanisms, and various personal *terminals.*

0.2 What is a Program?

The extreme speed of a computer requires that it be given in advance a plan for carrying out the calculations. In addition, the computer must carry out, also according to plan, the vast number of simple decisions that must be made. For instance, calculating payroll checks on the computer requires that the computer decide whether or not social security deductions shall be made in each of many thousands of cases. It is therefore necessary to supply the computer in advance with a complete set of instructions as to what calculations are to be performed and what logical decisions must be made. Such a set of instructions is called a *program.* Put another way, a program is a plan or "recipe" for going from the data of the problem to the required answers.

One important characteristic of a program is that it must be prepared in a language that the computer mechanism can "understand." The rules of grammar that are required by such languages are usually very detailed and must be followed exactly. The task of preparing, in precise form, programs for carrying out tasks on the computer is called *programming* and the practitioners of this skill are called *programmers.* The most successful programmers are those that can make sensible judgements about alternate approaches to a particular problem and select methods that are efficient in terms of both the persons that will be using the program and the utilization of the computer resources.

Because programming in machine language is so highly specialized, persons more familiar with other fields have previously found it difficult to translate their problems for solution on the computer. For this reason, various *problem-oriented* languages such as FORTRAN, ALGOL, and COBOL have been developed, ones whose rules of grammar are more readily learned and applied than those of a machine language.

A program prepared in one of these languages is first "translated" into the machine's language by a process called *compiling*, which is very similar to the translation of natural languages. After compilation, the translated version of the program is then acted on directly by the computer. It should be noted that the translation process is itself carried out by a complicated program called a *compiler*. The use of these languages, which are more oriented to the *users* of computers, have advanced enormously the application of computers and computer technology to many areas of business, engineering, and scholarly research.

0.3 What is BASIC?

While the advent of user-oriented and problem-oriented languages has been a great boon to the use of computers, most of these languages were designed either for the special use of certain groups of experts or at a time when relatively little was known about their translation. Hence, most of these languages are somewhat difficult to learn and then are somewhat difficult to apply. Recognizing that a great many potential computer users might find even the common user-oriented languages a barrier to intelligent use of computers, it was decided that a new simple language was needed. It must have very simple grammatical rules and must be capable of being learned in a very short time. The language BASIC thus came into being. It is very easy to learn, and can be applied to most computing problems (nonnumerical as well as numerical) quickly and easily. The purpose of this text is to provide a simple introduction to computer programming using the BASIC language, and then, in Part II, to develop a wide variety of applications involving relatively modest BASIC programs.

0.4 How a Computer is Used

As was previously stated, a computer must be supplied with a program and a set of data. In more specific terms, the program is supplied to the computer on some medium readable by the computer. Examples of such media are *punched cards* and *punched paper tape*. Another method of supplying the program is through a personal terminal device such as a Teletypewriter (TM). (Both Teletypewriter and Teletype are trademarks of the Teletype Corporation, but it is common to use the term "teletype" to refer to any typewriter-like terminal device.)

Computer systems differ not only in the way they receive the program, but also in the way they process the program and manage the use of their resources. Two common ways in which computer systems are organized are "batch processing" and "time sharing." Batch processing usually involves submitting the program to the computer in punched card or similar form, having the computer carry out the program when it comes its turn, and then printing the answers; these can be picked up later, usually at least several hours later. Time sharing involves submitting the program by typing it on a typewriter-like console, having the computer system "share" its resources by giving each program a small amount of attention when it needs it, and then providing the answers almost immediately on the same typewriter-like console. Time sharing also often provides for "storing" the program for the user; the next time it is needed it will be instantly available and not have to be retyped.

This distinction between batch processing and time sharing is necessarily oversimplified, as most computer systems today operate in a mode that has a combination of the properties of the two processes. Furthermore, certain types of applications are well handled in one type of computer environment, and poorly handled in the other. One type of application that time sharing does well is the training of inexperienced users. This is because the results of a student's program come back to him almost immediately—if his program works, he is "rewarded" immediately; if there are errors, he is told at once and can correct them before he forgets what he is doing.

Most of BASIC is as much at home in batch processing environments as it is in time sharing. However, BASIC does have the capability to be used in interactive programs, that is, programs that require the user's participation in order to achieve the desired results. For instance, playing a game of chess against the computer requires an interactive program. Interactive use of the computer can take place only in time sharing, which enables a program to communicate with a user instantly through a typewriter-like device connected directly to the computer system, perhaps through a telephone line. Many of the examples in this text can be used in either type of computer environment, but a few very important ones can be used only in a time-sharing environment.

1 Elementary BASIC

1.1 Introduction

The purpose of this chapter is to introduce the BASIC language. In many respects BASIC resembles other computer programming languages, but it is specifically designed to be simpler and easier to use. BASIC is also designed to facilitate communication between man and machine in a time-sharing system.

In most languages there is only a small amount of difference between what we must know in order to write simple programs and the additional details needed to organize complicated programs efficiently. That is, with most languages one must learn most of the details before writing even simple programs. In BASIC it is possible to write a tremendous variety of simple programs with very few different types of BASIC statements. In other words, BASIC facilitates the writing of simple programs. Surprisingly, as is shown in Part II, many quite substantial applications of the computer can be implemented through simple BASIC programs. BASIC also provides the power necessary even for the most sophisticated applications. We shall first introduce, by means of a very simple example, five of the most basic statements of BASIC.

1.2 A Very Simple Example

As our introduction to BASIC we select an example so simple that the calculational part can be immediately understood by all. (In fact, it is so simple that one may wonder why it is even a legitimate computer program.) The problem is to divide two numbers, say 147 and 69. A complete BASIC program for performing this task is the program DIVIDE, which is given below:

```
LIST

DIVIDE     30 AUG 70   20:54

100  READ  N, D
200  LET Q = N/D
300  PRINT N, D, Q
400  DATA  147, 69
500  END
READY
```

Before explaining this program, we show an actual computer run of it, which shows that the quotient of 147 and 69 is approximately 2.13043.

```
RUN

DIVIDE     30 AUG 70   20:55

 147            69              2.13043

TIME:  0.050 SEC.
READY
```

We now give an explanation of the five statements in this program. In line 100 is a READ statement. This causes two numbers to be taken or "read" from the data of the program (which appear in the DATA statement of line 400) and be assigned to variables whose names are N and D, respectively. That is, the first number read will be interpreted as the numerator and the second number will be the denominator. (Actually, BASIC does not care what names we pick for the variables in the program, but it is wise to choose names that can remind us what they stand for. Hence, we use N for numerator and D for denominator.)

Now that we have the numerator and denominator, we carry out the division in line 200, which is a LET statement. The result of the di-

vision is assigned to the variable Q, the quotient. The diagonal slash is the symbol for division, since the built-up fraction is difficult to type on a typewriter. The LET statement may be interpreted "LET the variable Q be equal to the value obtained by evaluating the expression or formula to the right of the '=' sign."

In line 300 the program instructs the computer to print three numbers, namely, the values of the numerator, the denominator, and the quotient. BASIC will print these numbers across the page on the same line. We elected to print not only the quotient Q but also the data to remind the user what his problem was.

Line 400 is a DATA statement, which supplies data to the READ statements in the program. Thus, to find the quotient of a different pair of numbers, all we must do is change the DATA statement and rerun the program. An END statement in line 500 merely serves to notify BASIC that the end of the program has been reached.

In summary, we observe that the BASIC program is made up of statements, each of which consists of a line number followed by an English word which gives the type of the statement. Most statements have other constituents such as arithmetic formulas, but a few, like END, are self-contained. Although the program DIVIDE is extremely simple, it embodies most of the ingredients that are common to all programs. The READ statement serves to supply the program with its data, which comes from the DATA statement. The LET statement causes calculations to be carried out. The PRINT statement causes the answer to be made known to the user. The END statement serves to indicate the end of the program.

1.3 Another Example

The conversion of distances from the metric system to the English system is an important problem, notwithstanding the expected shift to the metric system in England. Suppose that distances are given in meters and centimeters, and that we wish to convert them to feet and inches. One possible program for converting from meters and centimeters to feet and inches might be as follows:

1. Obtain the data in meters and centimeters.
2. Convert meters and centimeters to meters alone.
3. Convert meters to inches by multiplying by 39.37.
4. Determine the number of whole feet and the number of inches left over.
5. Print the answers.
6. Go back to step 1 for the next problem.

This program is perfectly adequate for a human computer with a little imagination. But, as we have seen, it must be more precisely stated in a computer language before the computer can be applied to the problem. The expression of this program in BASIC is given below under the name CONVRT.

```
LIST

CONVRT     30 AUG 70  20:55

100  READ M, C
110  LET MI = M + C/100
120  LET I = MI * 39.37
130  LET F = INT(I/12)
140  LET I = I - 12*F
145
150  PRINT M, "METERS,", C, "CENTIMETERS"
160  PRINT "CONVERTS TO"
170  IF F = 0 THEN 190
180  PRINT F, "FEET",
190  PRINT I, "INCHES"
200  PRINT
210  PRINT
220  PRINT
230  GO TO 100
235
240  DATA  1, 0
250  DATA  0, 2.54,  0, 60,  2, 5
999  END
READY
```

A computer equipped to understand BASIC will carry out the directions of this program, producing printed output as follows:

```
RUN

CONVRT     30 AUG 70  20:56

 1             METERS,        0              CENTIMETERS
CONVERTS TO
 3             FEET           3.37           INCHES


 0             METERS,        2.54           CENTIMETERS
CONVERTS TO
 0.999998      INCHES


 0             METERS,        60             CENTIMETERS
CONVERTS TO
 1             FEET           11.622         INCHES


 2             METERS,        5              CENTIMETERS
CONVERTS TO
 6             FEET           8.7085         INCHES


OUT OF DATA IN 100

TIME:  0.122 SEC.
READY
```

As with the program DIVIDE, the program CONVRT consists of statements, each one consisting of a line number, followed by an English word, followed in most cases by formulas or expressions. A detailed line-by-line discussion of this program is now given.

The first statement (100) is a READ statement. It causes the first two data appearing in the DATA statements (lines 240 and 250) to be assigned to the variables M and C, respectively. In this case, M and C were chosen to suggest meters and centimeters; X and Y or

any other letters could just as well have been used. Notice that the two variables are separated by a comma.

The LET statement (110) shows the principal way in which calculations are performed. The formula on the right-hand side of the equal sign is evaluated using the current values of the variables that appear. For the first problem, M and C have the values 1 and 0. After the LET statement in line 100 has been carried out, the variable M1 will have been given the value 1, which is the value of the expression 1 + 0/100. For the last set of data, where M and C are 2 and 5, respectively, M1 will be assigned the value 2.05, calculated from 2 + 5/100. Notice that '+' stands for addition and '/' for division, and that the division is performed before the addition, as in ordinary algebra.

In statement 120 the program converts meters to inches by multiplying by 39.37. This statement shows the use of the '*' to indicate multiplication, and also shows that ordinary decimal notation is used for numerical constants. The variable I is used to stand for inches.

The whole number of feet, denoted as F, in I inches is computed in line 130. First, I/12 is performed to determine the number of feet in I inches, including possibly a fractional part. We next employ a *function* to discard the fractional part and give the whole number of feet. The function here is named INT, which suggests its role—take the integer part of the argument. This function is one of several standard functions that are available in BASIC. In each case, the argument of the function is enclosed in parentheses following the three-letter name of the function.

Next, the number of inches left over is calculated in line 140. Starting with the original number of inches, we subtract the number of inches in F feet. The difference is the number of inches left over. Notice that *before* this statement is executed, the value of I is the total number of inches. It is this original or old value that is used in evaluating the expression I − 12 * F. This value is then assigned back to I. Thus, the new value of I, *after* statement 140 is performed, is the number of inches left over. This statement emphasizes that the LET statement is a command to perform calculations, and is not a statement of algebraic equality. The proper interpretation of this statement is "evaluate the formula on the right, and let this value be assigned to the variable on the left." Or, more specifically, "let I take on a new value found by subtracting 12 times the value of F from the old value of I."

At this point all the necessary calculations have been performed, and most of the rest of the program is devoted to printing out the answers. It is important that numerical answers be labeled so that the reader can make the proper interpretation. Statements 150, 160, 180, and 190 show that each number printed out is labeled, and also that it is important to print out the original data of the problem.

When the computer comes to perform statement 150, it will cause the value of M to be printed out in the first column, the word METERS to be printed in the second column, the value of C in the third, and the word CENTIMETERS in the fourth. The commas in the PRINT statement serve to form the columns, each of which is 15 characters (that is, letters, digits, special symbols, and spaces) wide. Any characters included between pairs of quotation marks become labels that are printed out exactly as they appear. Thus, the comma immediately following METERS lies inside the quotation marks and will be printed; the other commas will not be printed, since they serve only to separate the printed matter into columns.

Statement 160 is an example of a PRINT statement that contains only labeling information. The label, of course, will be printed at the beginning of the first column.

An extremely important requirement in most computer programs is the need to make decisions based on results of previous calculations. Usually, the program directs the computer to take one or the other of two paths in the program, depending on whether some relation is true or false. In BASIC the IF–THEN statement satisfies this need. Statement 170 specifies that if the variable F is, in fact, equal to zero (that is, if the relation "F = 0" is true), the computer should skip to statement 190. Otherwise, if F is not equal to zero (the relation "F = 0" is false), the computer continues to the next statement in sequence, which is numbered 180 in the example. The reason for using the IF–THEN statement at this point is to avoid printing '0 FEET' when in fact the number of feet is zero. A similar modification could also have been made to split up statement 150 so that '0 METERS' would not be printed when the number of meters is zero.

Statement 180 is again a PRINT statement, one that is executed each time the number of feet is greater than zero. It prints the value of the variable F in column one and the label "FEET" in column two. Notice the apparently redundant comma after the closed quote mark—this comma tells the computer to stay on the same line and await further printed information. If that comma were omitted, the next PRINT statement encountered would start printing on a new line in column one. The final comma permits the information in several separate PRINT statements to be printed on the same line.

Statement 190 prints the number of inches and the

label "INCHES". If there are zero feet, statement 190 starts printing in column one. If there are one or more feet, this statement starts printing in column three (which starts thirty spaces from the left margin), since columns one and two have been used by statement 180.

Statements 200, 210, and 220 contain a blank PRINT. The effect is to start a new line for each similar PRINT. In the example, the total effect is to skip three lines with the purpose of separating, for easy reading, the printing pertaining to the separate problems.

Statement 230 tells the computer to go to statement 100 instead of continuing in sequence. In this case, statement 100 is the start of the program, and another problem will be attempted.

Statements 240 and 250 are DATA statements. BASIC takes all the data appearing in all the DATA statements in the program and forms a (sometimes quite long) list of numbers. As READ statements are encountered, data are used consecutively from this list. If the data become exhausted, and a READ statement is then attempted, there will be produced the message "OUT OF DATA IN xxx," where xxx is the line number of the READ statement being attempted. In many problems, this is a normal way to reach the end; the computer keeps working problems until it runs out of data. In some cases, the "OUT OF DATA" message may mean that the program is wrong, since insufficient data may have been included.

It is important to realize that the number of data in each DATA statement is completely immaterial. Furthermore, a given READ statement can draw data from more than one DATA statement. Only the order of the data within the set of all DATA statements is important. However, it is useful for proofreading purposes to arrange the DATA statements neatly and in an orderly fashion.

Finally, every BASIC program must have an END statement. It must be the last (highest numbered) statement in the program; in this case it is statement 260. This statement marks the end of the program, and it is also used to stop computation.

The run shows that 1 meter converts to 3 feet, 3.37 inches (39.37 inches), and that 2.54 centimeters is almost, but not quite equal to 1 inch. Notice that in the second and third conversions, '0 METERS' is printed since the program does not provide for its omission (see Exercise 4).

The numerical answers printed by CONVRT show that BASIC prints numbers to no more than six significant digits, except for integers which are printed to the full accuracy of the computer. Less than six digits are used if possible, and trailing zeros are not printed. Numbers that happen to be integers appear without a decimal point. Thus '1.' means that the value was between .9999995 and 1.000005 and was rounded to 1. A '1' without a decimal point means that the computed value was exactly 1.

1.4 Some Additional Facts

We learn much about the BASIC language from inspecting the two example programs. For completeness, we now present certain facts and details about the language.

An important point about using a teletype terminal is that the letter "oh" must be visibly different from the numeral zero. Some terminals distinguish by slashing the letter "oh;" some terminals slash the zero (which is done in this book); some terminals type these two characters with obviously different shapes, such as a round circle for "oh" and narrow ellipse for the zero. Whatever system is used, the user should be aware that zero and "oh" mean entirely different things to the computer.

BASIC ignores spaces. Thus, statements 100 and 110 could just as well have been written

```
100READM,C
110L E TM 1=M+C/ 1 0 0
```

These two lines are difficult to read, and one should use spaces judiciously to promote easy reading. However, accidental omission or insertion of spaces causes no trouble. Spaces inside quote marks in PRINT statements, are, of course, not ignored.

BASIC also permits lines that are blank except for the line number, and examples of their use to separate parts of programs are shown in CONVRT. Line numbers so used remain legal line numbers for use in GO TO and similar statements.

BASIC allows variable names consisting of a single letter, or a single letter followed by a single digit. The variables used in the program DIVIDE are N, D, and Q, all single letters. In CONVRT we used M, C, M1, I, and F. Notice the M1 is the letter "em" followed by the numeral "one." The letter may be upper or lower case, for those users having an upper-case/lower-case terminal. In fact, all the official words of BASIC may be typed in either upper case or lower case or a mixture.

BASIC accepts numbers in the usual notation; that is, a string of digits possibly preceded by a plus or minus sign and possibly containing a decimal point. In addition, a given number can be "scaled" or multiplied by a power of ten by following the number by a letter

"E" followed by the power, which must be an integer number and may be either positive or negative. Thus, the following are legal numbers in BASIC:

12345 .12345 −123.45 +000.0005 −.012321 E-4

The last number is −.0000012321 since the E-4 tells us to multiply by ten-to-the-minus-four, which is .0001. Observe that E4 is not a legal number, because it is a legal variable name! Thus, any number containing the E notation must have at least one digit in front of the E.

The requirement that every formula must lie on a single line, with no superscripts, subscripts, or built-up fractions, calls for an understanding of how certain formulas are interpreted. In some cases, parentheses may be needed to clarify the meaning. For example:

Formula	BASIC
$A + \frac{B}{C}$	A + B/C
$\frac{A}{B + C}$	A/(B + C)
$\frac{AB}{CD}$	(A * B)/(C * D) or A * B/C/D
A − (B − C)	A − (B − C) or A − B + C
A^B	A ↑ B

Whenever you have several variables multiplied or divided, BASIC interprets them as if they were grouped from the left.

Formula	BASIC
((AB/C)/D)/E	A * B/C/D/E

A very common error is to use (−B + D)/2 * A for the formula

$$\frac{(-B + D)}{2A}.$$

This is wrong, and will be interpreted as

$$\frac{(-B + D)}{2} * A = \frac{(-B + D)A}{2}.$$

We should use

(−B + D)/(2 * A) or (−B + D)/2/A.

Most decisions in BASIC are made through the IF–THEN statement, which bases the decision on the truth or falsity of some relation such as "F = 0." Other more general relations may also be used, for example,

IF X + Y < 0 THEN 1000

would cause the program to jump next to statement 1000 if X + Y were in fact strictly negative at the time the statement is met. Other relational symbols and their meanings are:

<	less than
<=	less than or equal to
=	equal to
>	greater than
>=	greater than or equal to
<>	not equal to

In general, then, a relation consists of any arithmetic expression, followed by any relational symbol, followed by any arithmetic expression.

1.5 Summary

We have seen several types of BASIC statements. They are:

READ
DATA
LET
PRINT
GO TO
IF–THEN
END

READ and DATA are jointly used to enter numerical information into the computer. LET is used when computations must be performed. PRINT serves to make available the results by printing them. GO TO permits the program to "change course." IF–THEN allows the program to "change course" conditionally, according to the truth of some relation. END is the final statement in the program.

EXERCISES

1. Which of the following are legal variable names in BASIC? If not legal, state why.

A	B	AB
AA	AZ	Z23
K9	Z0	TO
7B	E2	X1

2. Which of the following numbers are legal in BASIC? If not legal, state why.

123.456E7	123.68E-3	−0.0001
321E-4	E2	−147
0E0	1E2.3	12345678765