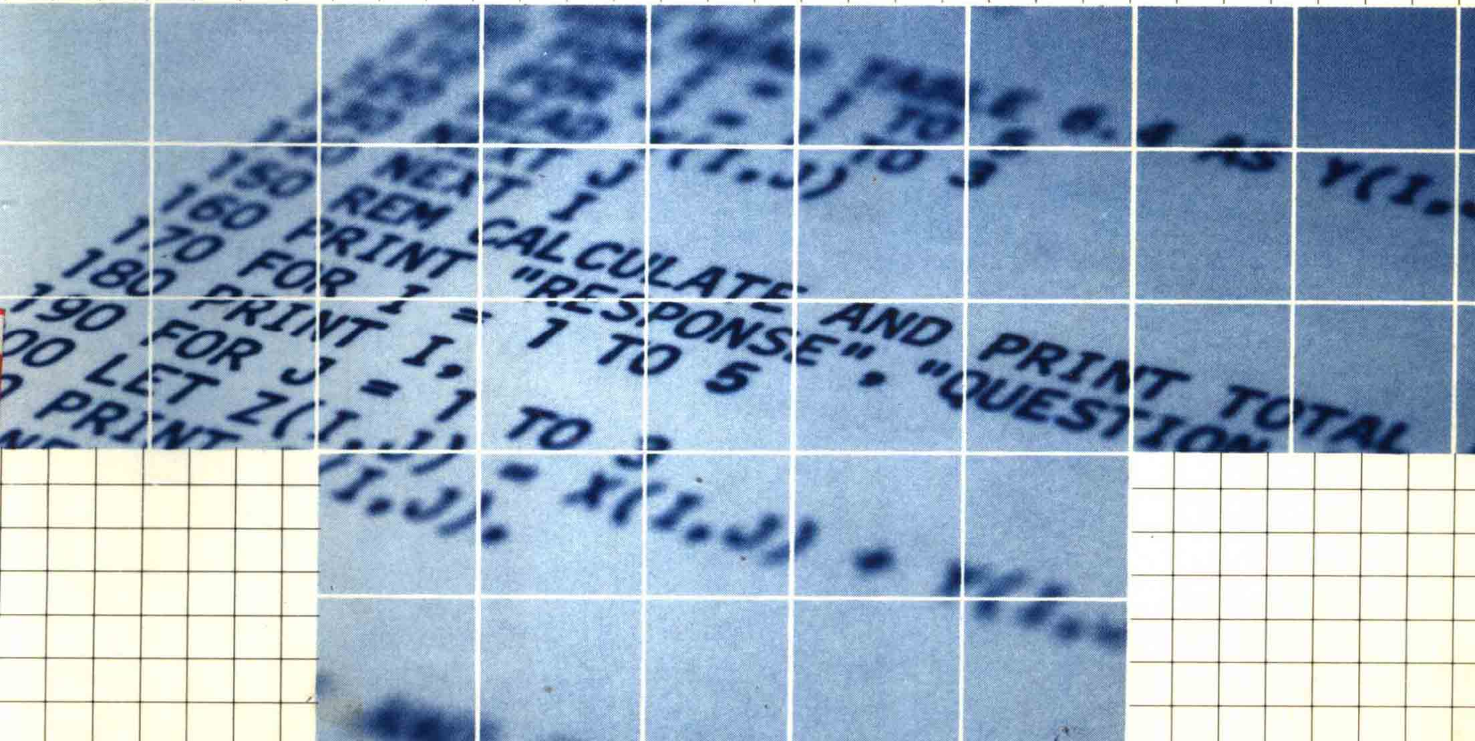
Essentials of BASIC Programming

Ralph M. Stair, Jr.

Ralph E. Janaro



```
150 REM CALCULATE AND PRINT TOTAL
160 PRINT "RESPONSE", "QUESTION
170 FOR I = 1 TO 5
180 PRINT I,
190 FOR J = 1 TO 3
```

Essentials of BASIC Programming

RALPH M. STAIR, JR.
The Florida State University

RALPH E. JANARO
Clarkson College of Technology

1984 • Revised edition

RICHARD D. IRWIN, INC.
Homewood, Illinois 60430

ISBN 0-256-02993-8

Library of Congress Catalog Card No. 83-81773

Printed in the United States of America

1 2 3 4 5 6 7 8 9 0 ML 1 0 9 8 7 6 5 4

Preface

The purpose of this text is to present the fundamentals of BASIC programming. Although this text is small and inexpensive, it covers major statements that can be found in books costing twice as much. This book has been designed to be used as a supplement to *Principles of Data Processing* by Stair. It can also be used with other books on computers and data processing or by itself for a course in BASIC programming.

The material has been organized into chapters that flow logically from one topic to the next. Before any new statement is introduced, students are shown why the statement is important and the types of problems that the statement can solve. This is followed by a clear explanation of the statement and one or more examples. In addition, each chapter contains several applications that reveal how a particular statement or a group of statements can be used to solve useful and realistic problems. Each chapter contains a number of questions. Each question has been designed to test a student's understanding of a particular statement or concept.

We would like to thank Claude McMillan and Robert Fetter for their comments and suggestions. In addition, we would like to acknowledge the support of Bill Anthony, the Department of Management, and the College of Business at Florida State University.

RMS
REJ

Contents

1

Introduction to program and application development

All of our lives have been altered as a result of the increased use of computer systems. Your checking account, phone bill, magazine subscriptions, and student records are examples. Unfortunately, at one time or another most of us have been victims of mistakes that occasionally occur in computer systems. It is unknown how many millions of dollars have been lost through these errors by companies of all sizes as well as governmental and nonprofit agencies at all levels, but the amount is large. You may be saying to yourself, "That cursed computer did it again," but these problems are not normally the fault of the computer equipent. You would not blame a hammer for a poorly driven nail, or a mixing bowl for a cake that had a horrible taste, and you should not blame the computer because you received the wrong phone bill or no magazine this month. Like the hammer or the mixing bowl, the computer is a tool—a tool we use to help us process data. The success or failure of a computer system to produce meaningful results does not, in general, depend on the computer equipment. It depends on the success or failure of people to use this electronic tool properly.

A computer program is the means by which people can control the computer to produce meaningful and beneficial output, but writing successful computer programs is not an easy task. How to write successful programs will be outlined in this chapter. To begin, we will look into what should be done before a program is written.

PREPROGRAMMING ACTIVITIES (PLANNING)

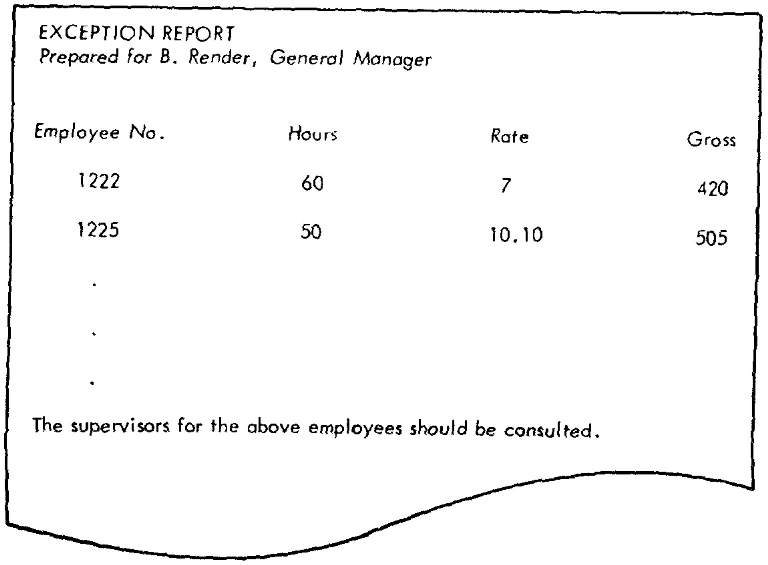
Some programming books lead you to believe that very little or nothing has to be done before introducing a program into a computer system; you just write the program on the back of an envelope or on a scrap piece of paper and then run it on the computer. Doing this for a program of any size would be like sketching a house on a soiled place mat or napkin at a restaurant and then beginning to construct the house without any blueprints or detailed plans. In both cases, the chances of success are small indeed.

Requirements

Computer programs are written because there is a problem to be solved or a need to be satisfied. Perhaps a medical clinic needs tighter controls on unpaid bills, or the police department has a problem in obtaining a fast and complete list of possible suspects for a crime, given partial descriptions such as a scar, foreign accent, nervous twitch in the left eye, hair color, height, and so on. Once the problem or need has been identified, the reports or information desired from the computer system, termed the *output requirements*, and the data needed to produce these reports, termed the *input requirements*, should be completely specified.

Requirements for Sound System, Inc. The general manager of Sound System, Inc., a producer of tweeters, midranges, woofers, horns, crossover systems, and other speaker parts, is concerned with rising payroll costs. After discussing the problem with the data processing department, the manager decided that a report should be prepared by the computer at 8:30 A.M. every Tuesday. The report should list all employees whose gross pay exceeded $400 for the previous week. To detail these needs to the data processing department, the general manager has drawn up a sample of the report that is to be generated by the computer system. It appears in Figure 1-1. This report is, then, the output requirement for this application.

FIGURE 1-1
Output requirements


EXCEPTION REPORT
Prepared for B. Render, General Manager

Employee No.	Hours	Rate	Gross
1222	60	7	420
1225	50	10.10	505
.			
.			
.			

The supervisors for the above employees should be consulted.

Now that the output has been specified, what input data is needed, and where is the data stored? Usually the data will be stored on disk, tape, or computer cards. We will assume that the data is already on computer cards from another program that prints the weekly pay checks; the last data card contains three zeros separated with commas to signify that there is no more data. Sample data appears in Table 1–1.

TABLE 1–1
Input requirements

Required data			*Employee number*	*Hours*	*Rate*
Variables to be used →			*N*	*H*	*R*
Sample DATA statements →	1000	DATA	1222,	60,	7
	1001	DATA	1223,	40,	5
	1002	DATA	1225,	50,	10.10
	.				
	.				
	.				
	9000	DATA	0,	0,	0

Analysis

Once the output and input requirements have been determined, the next step is to develop detailed plans that will be used in writing the program. Like a blueprint for a house, flowcharts are used in specifying how the computer program is to be written or constructed. This analysis also reveals how the input data is to be processed to get the desired reports.

Analysis for Sound System, Inc. The first step is to develop a flowchart for the application. Some of the symbols used in application flowcharting are on page 4. A complete set of symbols used in application and program flowcharting can be drawn with the aid of the template shown in Figure 1–2.

The purpose of an application flowchart is to use symbols to give a general picture of what functions the computer system should perform. The application flowchart reveals how many programs are to be written, what reports or documents are to be produced, and the form and source of the input data.

The application flowchart for Sound System, Inc., is straightforward. The program should read data from a card file and print a document containing a list of all employees earning over $400 per week. See Figure 1–3.

The application flowchart in Figure 1–3 gives an overall picture of the application. But how should the program be designed and written? We must analyze the exception report program in more detail. Like the application flowchart, the program flowchart uses symbols to describe computer actions. Some of these symbols are on page 5.

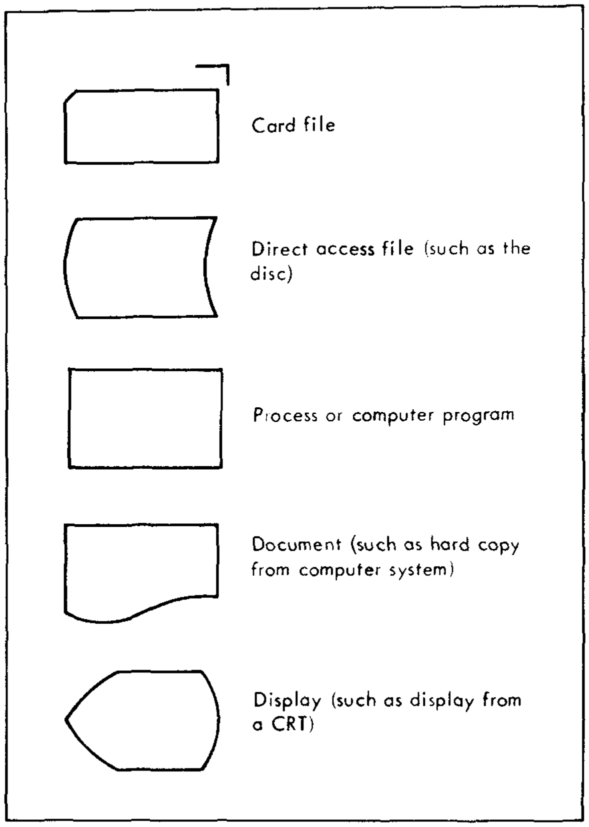
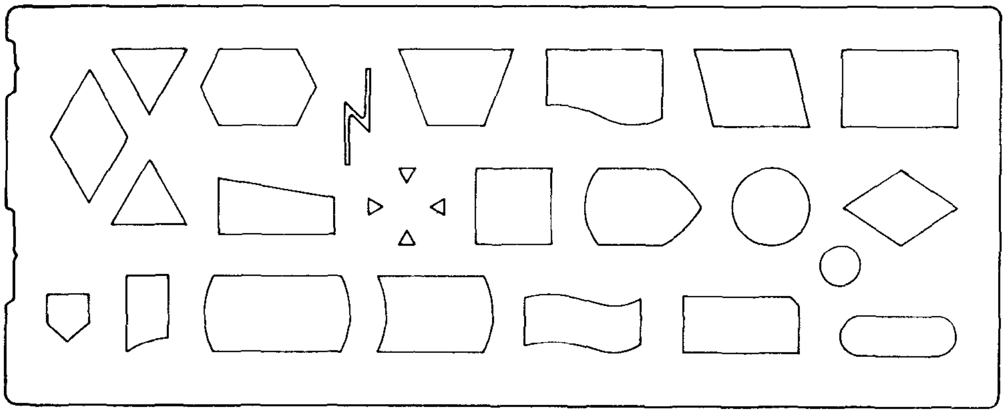
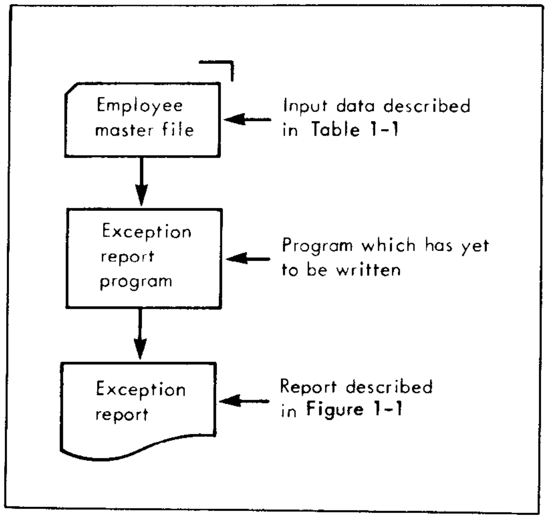

Card file
Direct access file (such as the disc)
Process or computer program
Document (such as hard copy from computer system)
Display (such as display from a CRT)

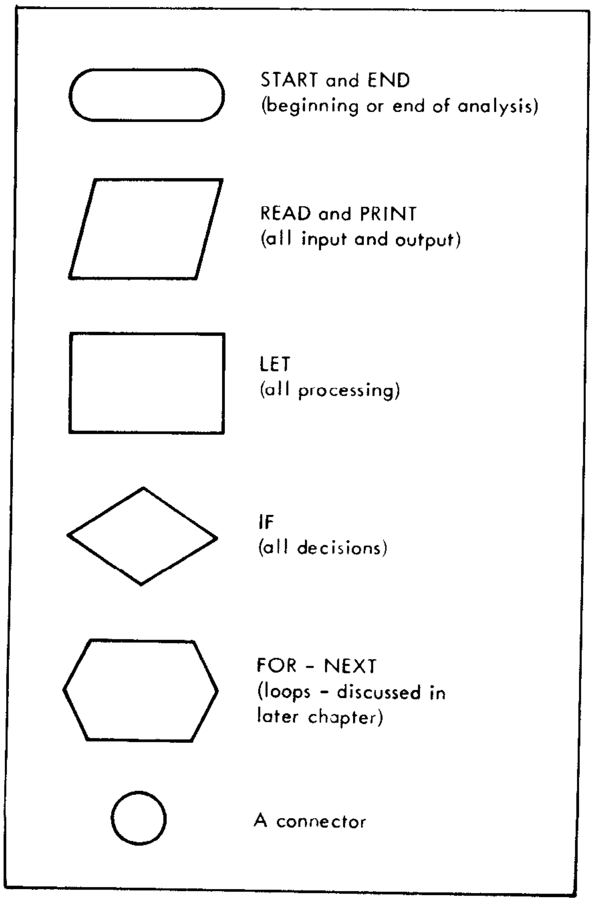
FIGURE 1-2
Template for drawing symbols used in application and program flowcharting.



Source: This material is reproduced with permission from American National Standard X3.5—1970. © 1970 by the American National Standards Institute. Copies may be purchased from the American National Standards Institute, 1430 Broadway, New York.

FIGURE 1-3
Application flowchart for Sound System, Inc.


Employee master file
Input data described in Table 1-1
Exception report program
Program which has yet to be written
Exception report
Report described in Figure 1-1


START and END (beginning or end of analysis)
READ and PRINT (all input and output)
LET (all processing)
IF (all decisions)
FOR - NEXT (loops - discussed in later chapter)
A connector

Now, the next step is to use these symbols to construct a program flowchart. But how is this done? One approach is to think about how you would generate the exception report by hand, using pencil and paper. Here is what you might do:

1. Start the analysis.
2. Print the headings for the exception report.
3. Read N, H, R (the employee number, hours worked, and rate).
4. Is N = 0? (When N is 0, there is no more data.)
   a. No—go to the next statement, which is number 5.
   b. Yes—go to statement 8, which prints the final message.
5. Let G be equal to H times R. This calculates the gross pay.
6. Is G less than or equal to $400 for this employee?
   a. Yes—go to statement 3 and process another employee.
   b. No—go to the next statement, which is number 7.
7. Print N, H, R, G. (Print the employee number, hours, rate, and gross pay.) Then go to statement 3 and process another employee.
8. Print the final message, which is "THE SUPERVISORS FOR THE ABOVE EMPLOYEES SHOULD BE CONSULTED."
9. End the analysis.

You may have done this in a different manner, but the results would have been the same. Now, we will develop the program flowchart. For your convenience, the above numbers have been included (see Figure 1–4).

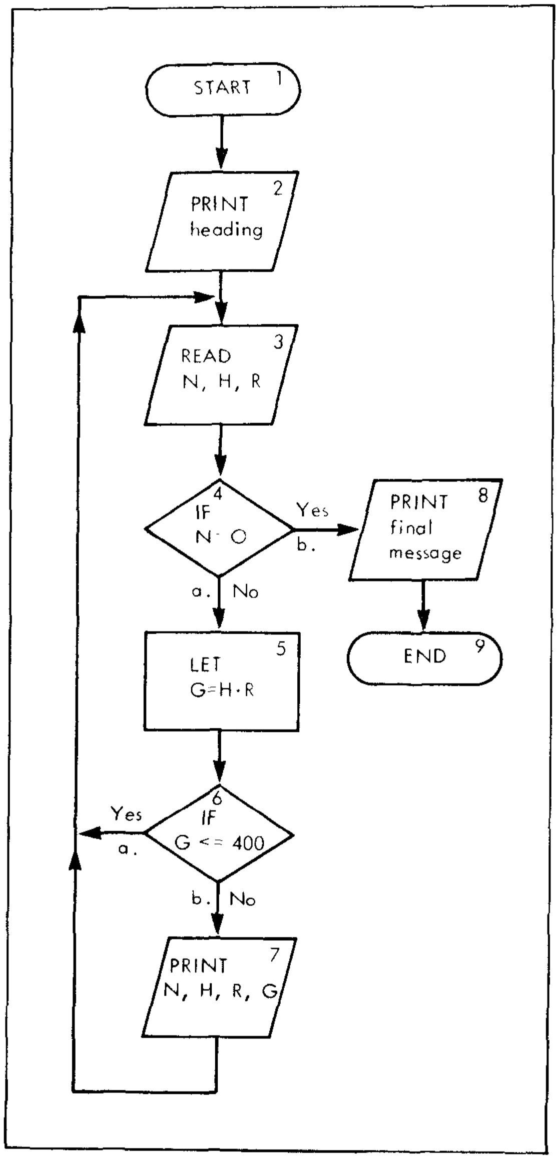
The flowchart in Figure 1–4 graphically displays the same information as described above. After the headings such as "EXCEPTION REPORT" are printed, N, H, and R are READ. If N = 0, then there are no more data, and the program stops after printing a brief message. Otherwise gross pay, G, is calculated, and checked to see if it is greater than 400. If G is greater than 400, N, H, R, and G are printed as requested by the general manager, after which the program instructs the computer to loop back and process the data for the next employee. Of course, if G is not greater than 400, the program instructs the computer to loop back to process the data for the next employee without any printing.

PROGRAMMING

Writing the program is the next step. Because this will be the subject of the following chapters, a detailed discussion will not be given here. It should be mentioned, however, that the program should be as simple and as straightforward as possible. This will make modifying the program later, if it is needed, easier and less confusing.

The program for Sound System, Inc. Using all the documents developed in the preprogramming steps, a programmer in the data processing department was able to write the necessary program. The program is shown on page 8.

FIGURE 1-4
Flowchart for Sound System, Inc.



It must be stressed that you are not expected to understand this program. It is only given here to show the natural progression of developing a successful computer program. As you look at the instructions in the program, however, you should note the similarity between it and the program flowchart previously given. You should also recognize the orderly progression from the written instructions, to the flowchart, to the program itself.

PROGRAM 1–1

```
10 PRINT "EXCEPTION REPORT"
20 PRINT "PREPARED FOR B. RENDER, GENERAL MANAGER"
30 PRINT
40 PRINT "EMPLOYEE NO.","HOURS","RATE","GROSS"
50 READ N,H,R
60 IF N = 0 THEN 110
70 LET G = H * R
80 IF G< = 400 THEN 50
90 PRINT N,H,R,G
100 GO TO 10
110 PRIMT "THE SUPERVISOR FOR THE ABOVE"
120 PRINT "EMPLOYEES SHOULD BE CONSULTED"
1000 DATA 1222,60,7
1001 DATA 1223,40,5
1002 DATA 1225,50,10.10
9000 DATA 0,0,0
9999 END
```

POST-PROGRAMMING ACTIVITIES

Many mishaps and computer system errors are caused by not completely performing postprogramming activities. The following activities are easy to perform, and they can prevent untold problems.

Testing

Every statement must be tested to assure that there are no programming mistakes (called "bugs"). Test data must be developed that will test every part of the program. If this is not done, a mistake in a program may go undetected for months. You may hear someone from the data processing department screaming with panic, "The program developed a 'bug' today."

Of course, the mistake didn't magically develop in the program after it was written. The mistake was always in the program, but the part of the program containing the mistake may never have been tested. Thus it is important to develop test data that will test *every* statement in the program.

Program testing for Sound System, Inc. The following output was obtained from the program.

OUTPUT FROM PROGRAM 1–1

```
? INITIAL PART OF STATEMENT NEITHER MATCHES A STATEMENT KEYWORD NOR HAS
A FORM THAT IS LEGAL--CHECK MISSPELLING IN LINE 110
```

As seen from the output, some type of mistake has been made in line **110**. Look at line **110** in Program 1–1. The keyword "**PRINT**" has been misspelled. This type of error is a grammatical mistake, which results in an error message. When line **110** is retyped and the mistake corrected, the program gives the following output from the computer.

As you can see, there is still something wrong with the computer program. The headings should be printed only once. Look at *line 100* in the program, and compare it to the flowchart. Do you see the problem? Line **100** directs the computer to **GO TO 10**, where the headings

OUTPUT FROM PROGRAM 1–1

```
READY
RUNNH


EXCEPTION REPORT
PREPARED FOR B. RENDER, GENERAL MANAGER

EMPLOYEE NO.  HOURS          RATE          GROSS
 1222          60             7             420
EXCEPTION REPORT
PREPARED FOR B. RENDER, GENERAL MANAGER

EMPLOYEE NO.  HOURS          RATE          GROSS
 1225          50             10.1          505.
EXCEPTION REPORT
PREPARED FOR B. RENDER, GENERAL MANAGER

EMPLOYEE NO.  HOURS          RATE          GROSS
THE SUPERVISOR FOR THE ABOVE
EMPLOYEES SHOULD BE CONSULTED
```

are printed. Line 100 should be GO TO 50 to process another employee. When this correction is made, the program will work. See Program 1–2.

Reflect a moment. If the test data did not include an employee whose gross pay exceeded $400, this error would not have been detected because the statement in line 80 would always send the computer to line 50, and the statements in lines 80 and 90 would not have

PROGRAM 1–2

```
10 PRINT "EXCEPTION REPORT"
20 PRINT "PREPARED FOR B. RENDER, GENERAL MANAGER"
30 PRINT
40 PRINT "EMPLOYEE NO.","HOURS","RATE","GROSS"
50 READ N,H,R
60 IF N = 0 THEN 110
70 LET G = H * R
80 IF G< = 400 THEN 50
90 PRINT N,H,R,G
100 GO TO 50
110 PRINT "THE SUPERVISOR FOR THE ABOVE"
120 PRINT "EMPLOYEES SHOULD BE CONSULTED"
1000 DATA 1222,60,7
1001 DATA 1223,40,5
1002 DATA 1225,50,10.10
9000 DATA 0,0,0
9999 END

READY
RUNNH


EXCEPTION REPORT
PREPARED FOR B. RENDER, GENERAL MANAGER

EMPLOYEE NO.  HOURS          RATE          GROSS
 1222          60             7             420
 1225          50             10.1          505
THE SUPERVISOR FOR THE ABOVE
EMPLOYEES SHOULD BE CONSULTED
```

been tested. *Thus it is important to prepare test* data that tests *every* statement in the program.

Implementation

The final step is to *phase in* the use of this program to produce the exception report that is to be presented to the general manager every Tuesday morning. If this report is now being produced manually, then both the manual report and the computer-prepared report should be run together in *parallel* until everyone is confident that the computer is preparing the report as expected. Then the manual report may be *phased out;* the new computer program is now running smoothly and providing the desired information to the general manager.

SUMMARY

Great speeches have been scribbled on the back of envelopes, but, in general, successful computer programs require more planning. Steps in writing a program are like links in a chain. If one is weak or absent, there will be trouble. In computer programming, what *can* go wrong *will* go wrong. In this chapter, the minimum requirements for building a successful computer program have been outlined. These activities were:

1. Determine the *output requirements.*
2. Determine the *input requirements.*
3. Conduct the *analysis* and construct a *flowchart.*
4. Write the *program.*
5. *Test* the program and make appropriate corrections.
6. *Implement* the program.

You may have been wondering what should be done when several programs interact, and the output from one becomes the input to another. The procedures outlined in this chapter should be logically expanded to analyze the entire system of programs. Starting with definition of output requirements and ending with implementation, the entire system of programs should be considered in addition to each individual program in the system. The procedures outlined do not substantially change when more complex applications are developed.

Sound System, Inc. summary. To bring meaning and realism to the procedures mentioned in this chapter, an application for Sound System, Inc., was examined. It was assumed that the data was on computer cards and that this data was used with a different program to produce weekly paychecks. Of course, the data could be on disk or magnetic tape, and the program discussed in this chapter may be one of several programs that comprise a *system of programs that write the* checks and produce quarterly reports, W-2 forms, and other reports.

It should also be mentioned that *special forms exist to help in spec-*

ifying output and input requirements, developing flowcharts, and writing computer programs.

QUESTIONS

1–1. What is the cause of most computer-related mistakes and problems?

1–2. What is involved in determining output requirements?

1–3. What is the purpose of testing? How should it be accomplished?

1–4. Once a new program has been written and tested, should the old program that it replaces be immediately abandoned? Why or why not?

2

Introduction to BASIC

ADVANTAGES OF BASIC

One reason that BASIC is so popular is its conversational nature. A BASIC program is very similar to the instructions you would write for a person. BASIC makes communicating with a computer natural, simiple, and straightforward.

Another advantage of BASIC is its many built-in conveniences. Handling a large table of numbers can be very difficult in other programming languages. In BASIC you can command the computer to print a table of more than 100 numbers or names with one simple instruction. In addition, newer versions of BASIC have excellent file-handling capabilities.

To get you started, here are some BASIC computer programs. Do not try to understand the details. Relax and try to get the flavor of programming.

Application 2-1. BASIC can be used like a hand calculator. If you want to find the circumference of a circle with a diameter of four inches, you multiply 4 times a constant π, which is 3.14159. The following BASIC program will make this calculation:

Application 2-2. You can also instruct the computer to read data

PROGRAM 2-1

This is the program that you type.

```
10 LET C=4*3.14159
20 PRINT C
30 END
RUNNH
```

RUNNH means run with no heading. You type this.

```
 12.5664
```

This is the output from the computer. It is printed after you type RUNNH and hit the return key.