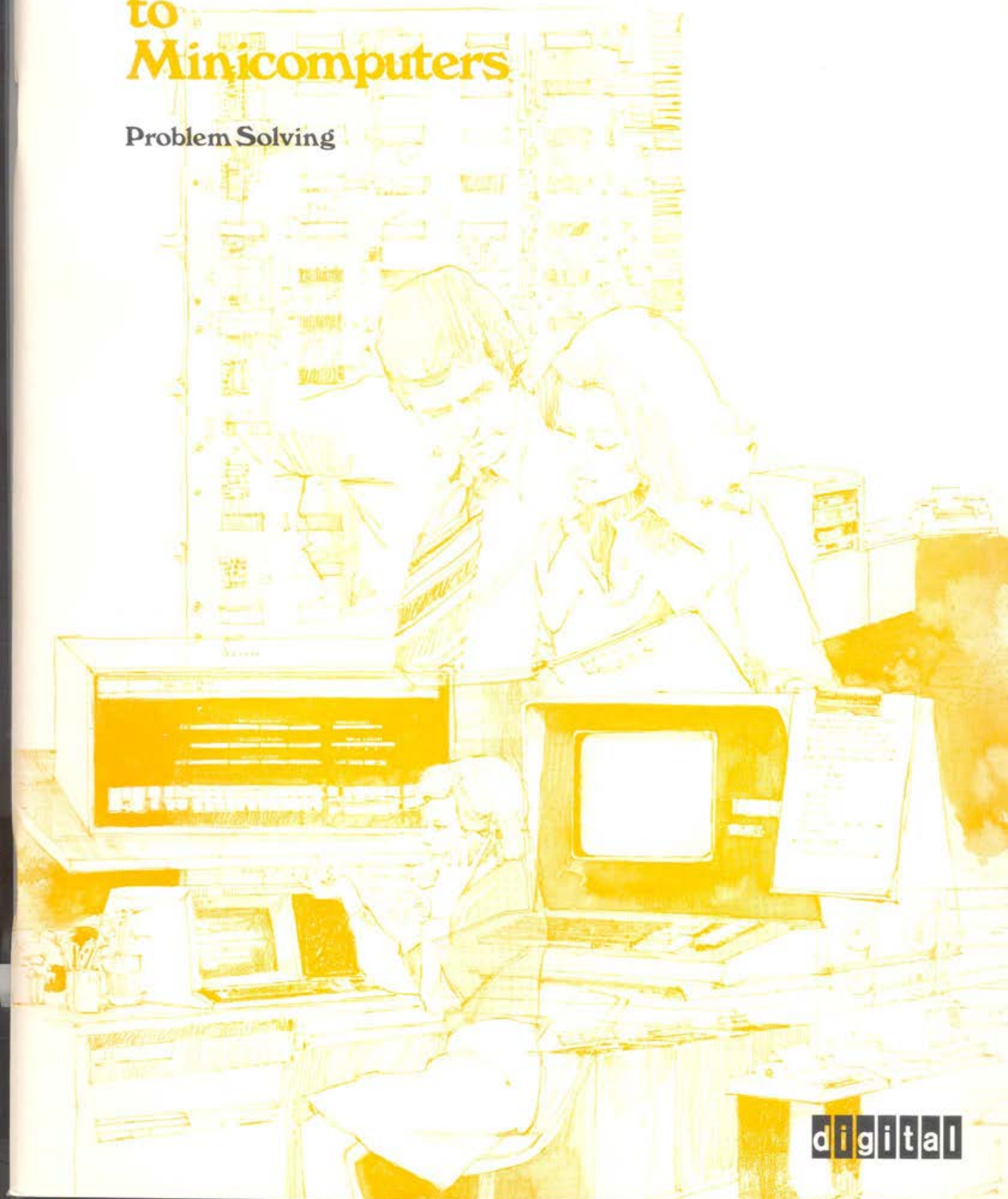


# Introduction to Minicomputers

Problem Solving



digital

1st Printing, June 1976  
2nd Printing (Rev), October 1977  
3rd Printing, August 1979

Copyright © 1976, 1977, 1979 by Digital Equipment Corporation

The reproduction of this workbook, in part or whole, is strictly prohibited. For copy information contact the Educational Services Department, Digital Equipment Corporation, Bedford, Massachusetts 01730.

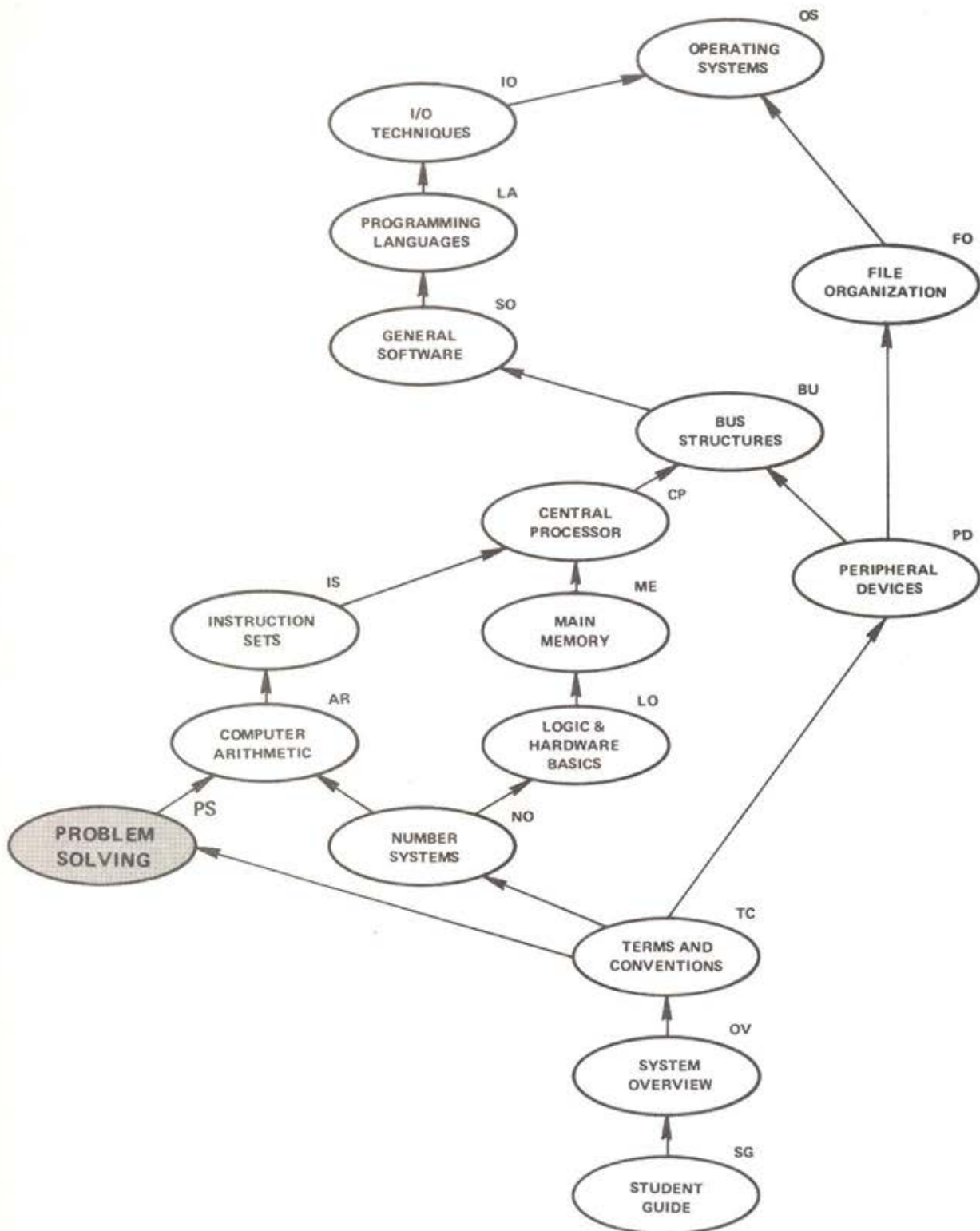
Printed in U.S.A.

**INTRODUCTION TO MINICOMPUTERS**

## **Problem Solving**

### **Student Workbook**

# COURSE MAP



## CONTENTS

<b>Introduction</b> .....	1
<b>Flowcharting Concepts</b> .....	3
Objectives and Sample Test Items .....	3
Phases of Problem Solving .....	7
Flowchart Symbols .....	7
Exercises and Solutions .....	13
<b>Flowcharting</b> .....	21
Objectives and Sample Test Items .....	21
Exercises and Solutions .....	27
<b>Steps in Problem Solving</b> .....	37
Objective and Sample Test Items .....	37
Define the Problem .....	39
Plan the Solution .....	40
Flowchart .....	41
Code the Program .....	42
Translate .....	42
Debug .....	42
Document .....	43
Exercises and Solutions .....	45

# Problem Solving

## Introduction

Computers are like people; they get their jobs done one step at a time. Computers, however, must be *programmed* as to where to start, stop, and what to do at each step along the way. People can decide for themselves what must be done if they stop and think.

Problem solving is the art of breaking down huge, complex processes into sequences of small, performable steps. When each step in a complex process is defined, a computer can be programmed to perform the process.

The first lesson in this module describes the phases of problem solving and explains a set of symbols that aid in depicting the flow of steps to be followed in performing a task. The second lesson shows how large tasks can be broken down and ordered using these symbols. The third and last lesson in this module discusses what happens in each problem-solving phase when computers are used to perform the steps of the task.

## Flowcharting Concepts

### OBJECTIVES

1. Given the terms "algorithm" and "initialization" and four definitions for each term, be able to select the correct definition for each term.
2. Given a list of activities, be able to label the four activities that constitute the phases of problem solving.
3. Given a series of flowchart symbols and lists of symbol names and functions, be able to match each symbol with its name and function.
4. Given a sample flowchart, be able to label each branch.
5. Given a sample flowchart that includes two loops, be able to label the steps within the range of each loop.

### SAMPLE TEST ITEMS

1. Circle the letter of the definition of each of these terms.

#### Algorithm

- a. Program written in a language the computer can translate.
- b. Sequence of steps to be followed in performing a task.
- c. Sequence of steps for correcting program errors.
- d. Sequence of symbols that represents a problem to be solved.
- e. None of the above.

## SAMPLE TEST ITEMS


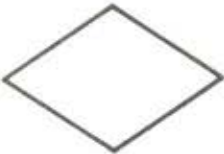
### Initialization

- a. Outlining the four phases of problem solving.
  - b. Defining the steps to be followed in executing the program.
  - c. Setting up the symbolic representation of each step of the program.
  - d. Setting up beginning conditions.
  - e. None of the above.
2. In the column labeled Phase of Problem Solving, write a T if the statement is a phase of problem solving. Write an F if the statement is not a phase of problem solving.

Statement	Phase of Problem Solving
Prepare a flowchart.	_____
State beginning values of variables.	_____
Define the problem.	_____
Identify resources.	_____
Choose proper computer language for the task.	_____
Determine project costs.	_____
Estimate program length.	_____
Analyze the task.	_____
Analyze the data output.	_____
Verify the steps.	_____

### SAMPLE TEST ITEMS

3. Basic flowchart symbols, their names, and their functions are given below. In the column labeled Name, write the letter that identifies the symbol's name. In the column labeled Function, write the letter that identifies the symbol's function.

Symbol	Name	Function
	_____	_____
	_____	_____

#### Names

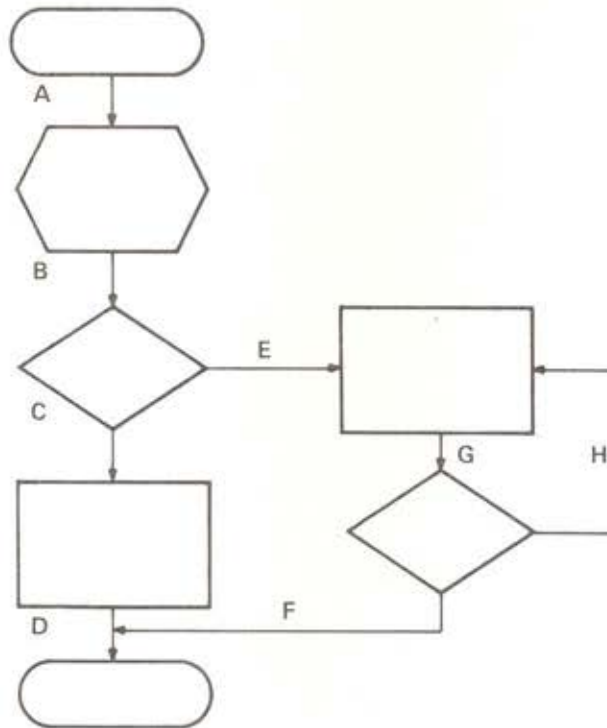
- a. The process symbol
- b. The decision symbol
- 
- 
- 

#### Functions

- i. Indicates that a choice must be made as to what to do next.
- j. Symbolizes the steps that process data.
- 
- 
-

# SAMPLE TEST ITEMS

4. Each arrow in the flowchart below is labeled with a letter. In the answers spaces provided, write the letters of the arrows that represent flowchart *branches*. (Note: You will not need all the answer spaces provided).



Answers: \_\_\_\_\_

Mark your place in the workbook and view Lesson 1 in the audio-visual presentation, "Problem Solving."

## Phases of Problem Solving

Problem solving consists essentially of devising a sequence of steps, called an *algorithm*, to be followed in performing a task. Problem solving is basically a 4-phase process:

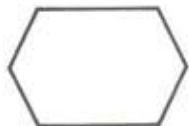
1. *Define the problem* – describe the problem: the input and the desired output.
2. *Identify resources* – determine what you will be using to do the job.
3. *Analyze the task* – break it down into small sequential steps.
4. *Verify the steps* – perform each step to prove that the sequence solves the problem.

## Flowchart Symbols

One of the most useful tools in problem solving is the *flowchart*. A flowchart is a series of standard symbols that pictorially represent the steps of an algorithm. The most important flowchart symbols used by programmers are:



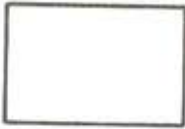
The *terminal* symbol, which is used to mark both the *beginning* and the *end* of the algorithm.



The *initialization hexagon*, which represents beginning conditions, such as the starting values of variables.



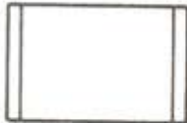
The *I/O parallelogram*, which is used to represent both input and output operations.



The *process rectangle*, which symbolizes the steps in which sequential actions occur.



The *decision diamond*, which indicates what to do next, provided a condition is met or not met.



The *predefined process* symbol, which can be used for drawing clearer flowcharts and saves a great deal of writing. It is used when a sequence of steps must appear several times in a flowchart. You draw the sequence only *once*, with a descriptive title. From then on, you can use the predefined process symbol to represent the entire sequence where it is necessary.



The *arrow*, which joins symbols and indicates the sequence in which steps are to be performed.



The *connector*, which has the same function as the arrow. Connectors are used to simplify complex flowcharts with many branches. In addition to simplifying a flowchart, connectors also clarify it. For example, compare the flowcharts shown in Figures 1 and 2.

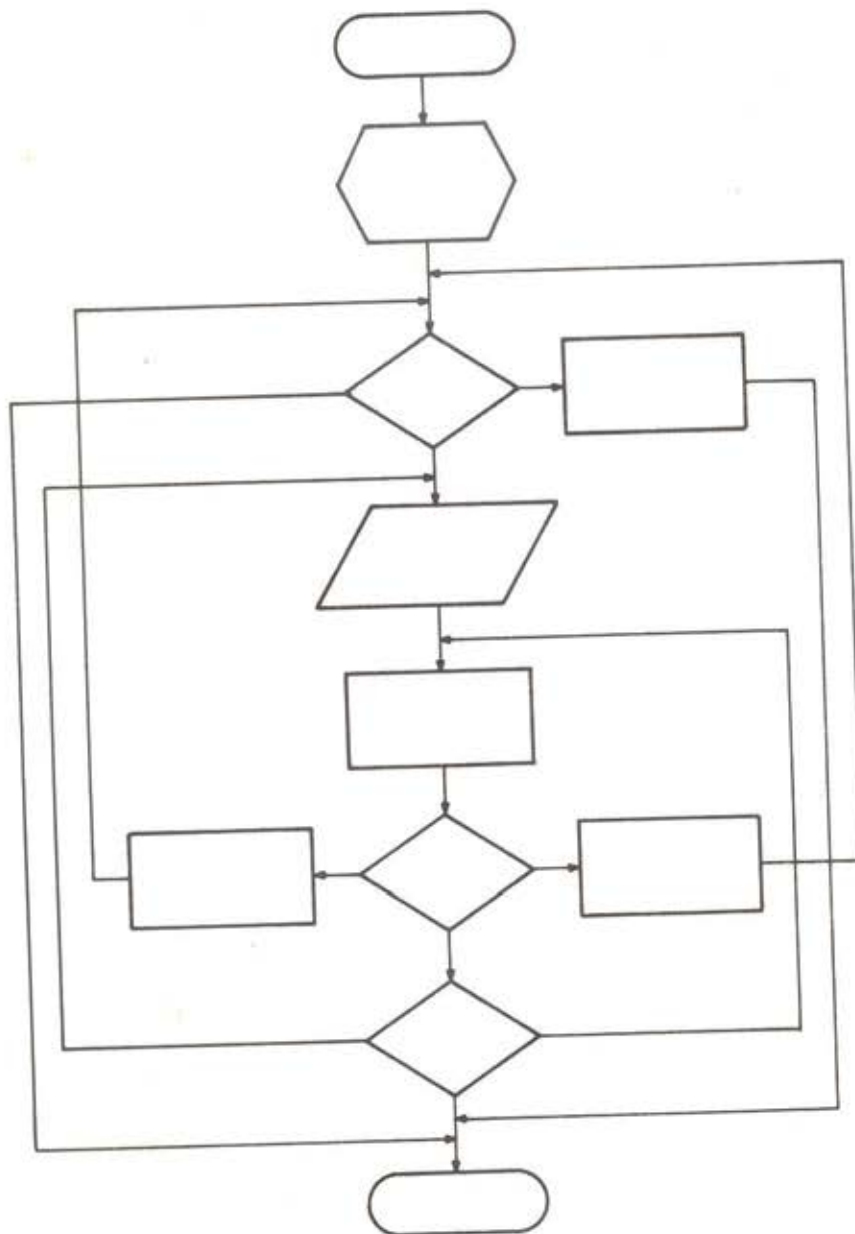


Figure 1 Flowchart Using Arrows Only

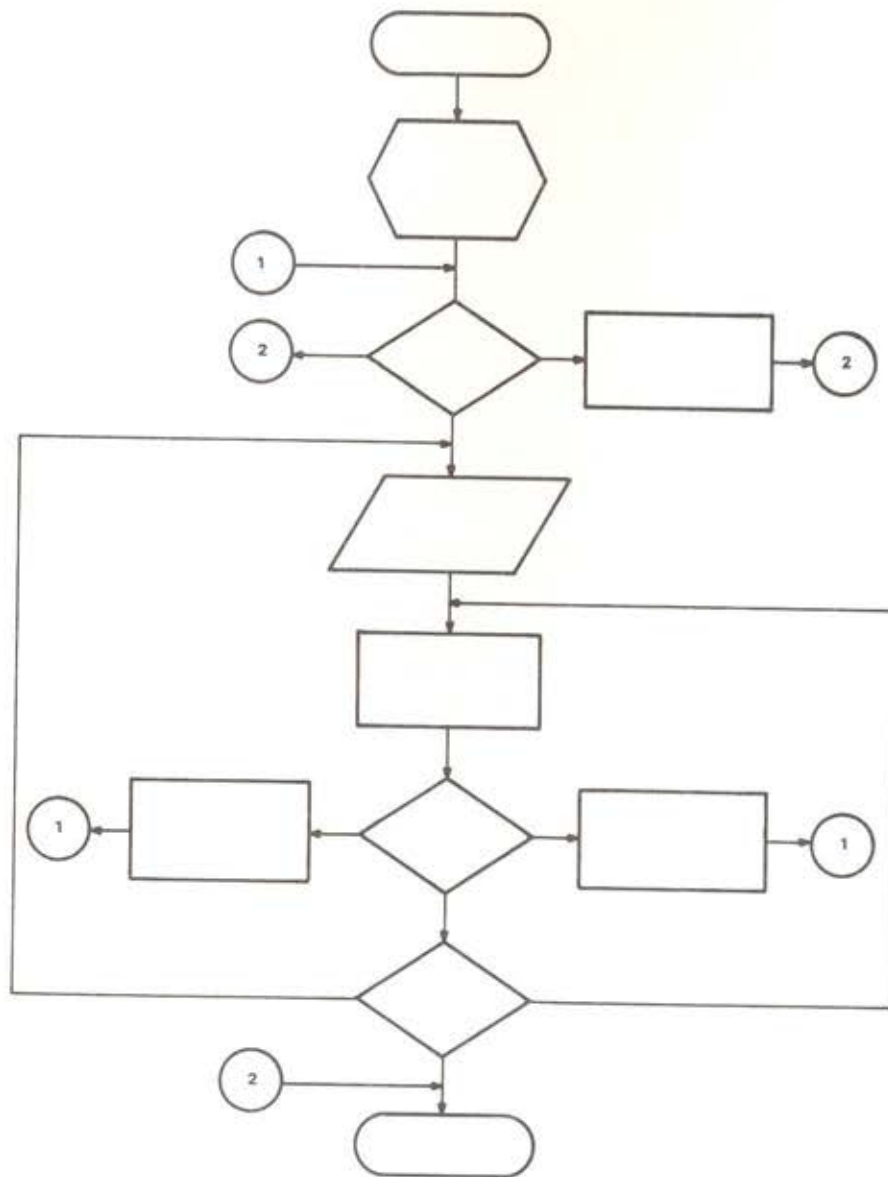


Figure 2 Flowchart Using Arrows and Connectors

As shown in Figure 2, an identifier, usually a letter or number, is written inside each connector. Sequence flows from one connector to the other with the same identifier.

You have now been introduced to the major flowchart symbols. Using these symbols, you can draw an endless variety of flowcharts anytime you need to define and describe a task. However, nearly all the flowcharts you draw will have certain things in common, such as *loops* and *branches*.

- **Branches** – Indicate alternative steps to be performed in a sequence. See Figure 3.
- **Loops** – Describe processes to be repeated until some condition is met. See Figure 4.

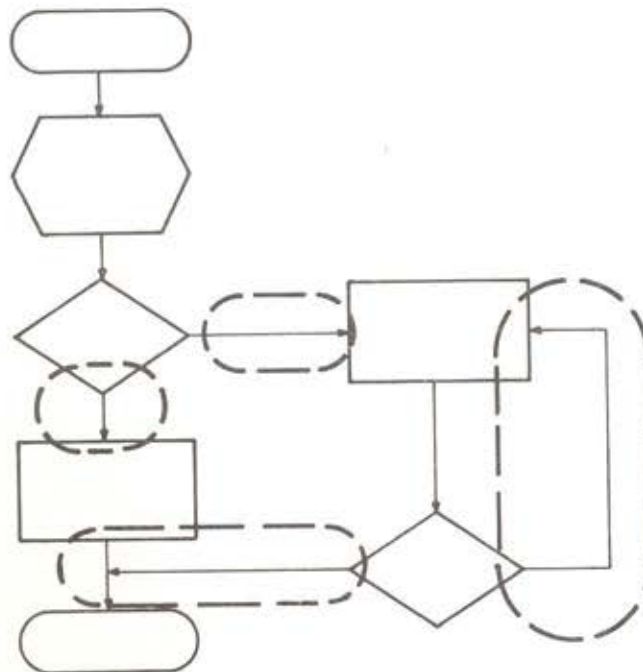


Figure 3 Branches

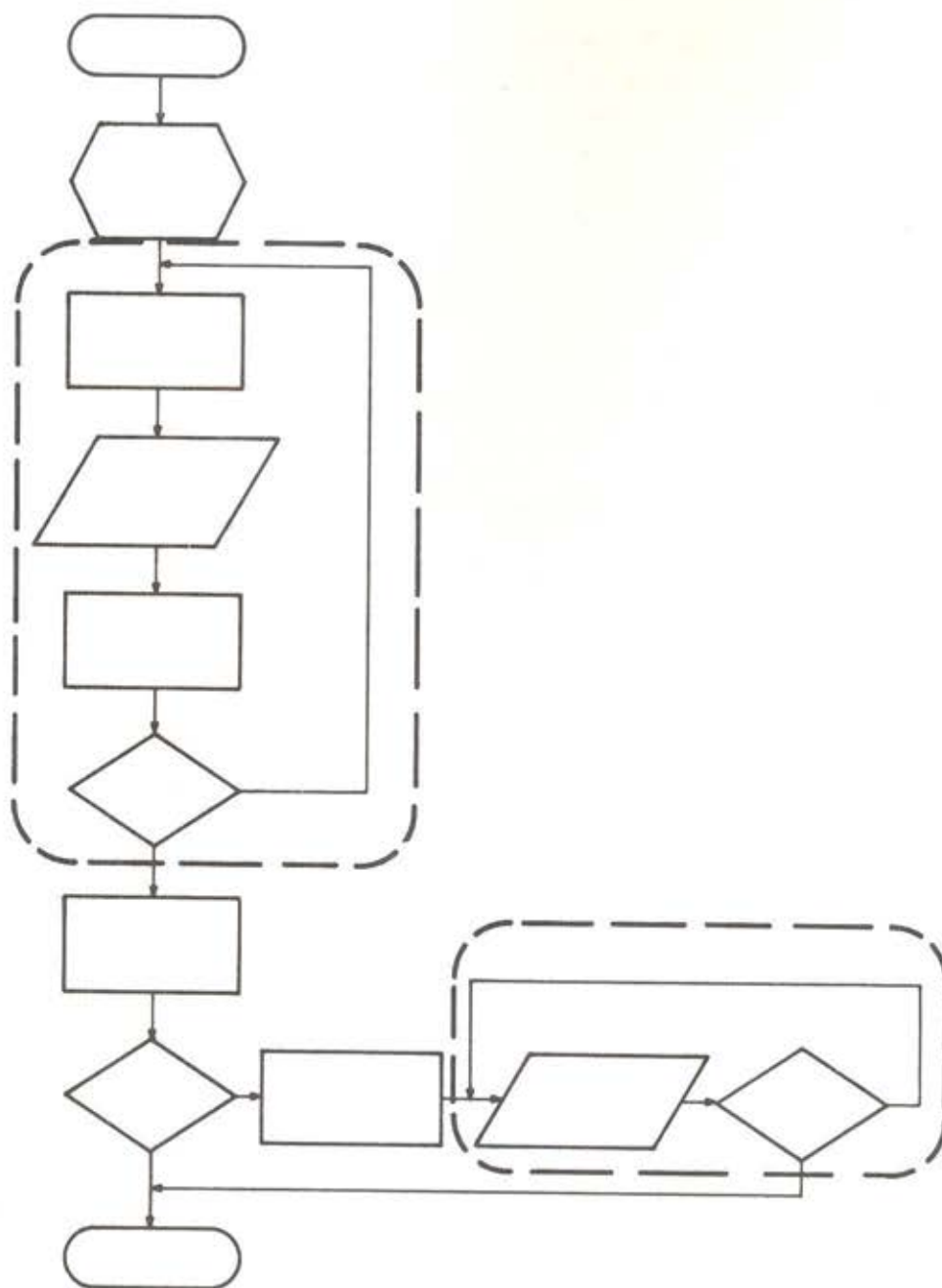


Figure 4 Loops

---

## EXERCISES

---

1. List the four phases of problem solving.
  
  
  
  
  
  
  
  
  
  
2. Define the following terms:
  - a. Algorithm
  
  
  
  
  
  
  
  - b. Initialization

---

## SOLUTIONS

---

1. The four phases of problem solving
  - a. Define the problem.
  - b. Identify resources.
  - c. Analyze the task.
  - d. Verify the steps.
2. Definitions
  - a. **Algorithm** – A sequence of steps to be followed in performing a task.
  - b. **Initialization** – Setting up beginning conditions.

---

## EXERCISES

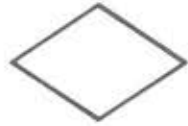
---

3. Write the name and use of each of the following flowchart symbols:

a.



b.



c.



d.



e.



f.



g.



h.

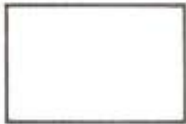



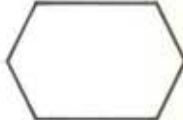





---

## SOLUTIONS

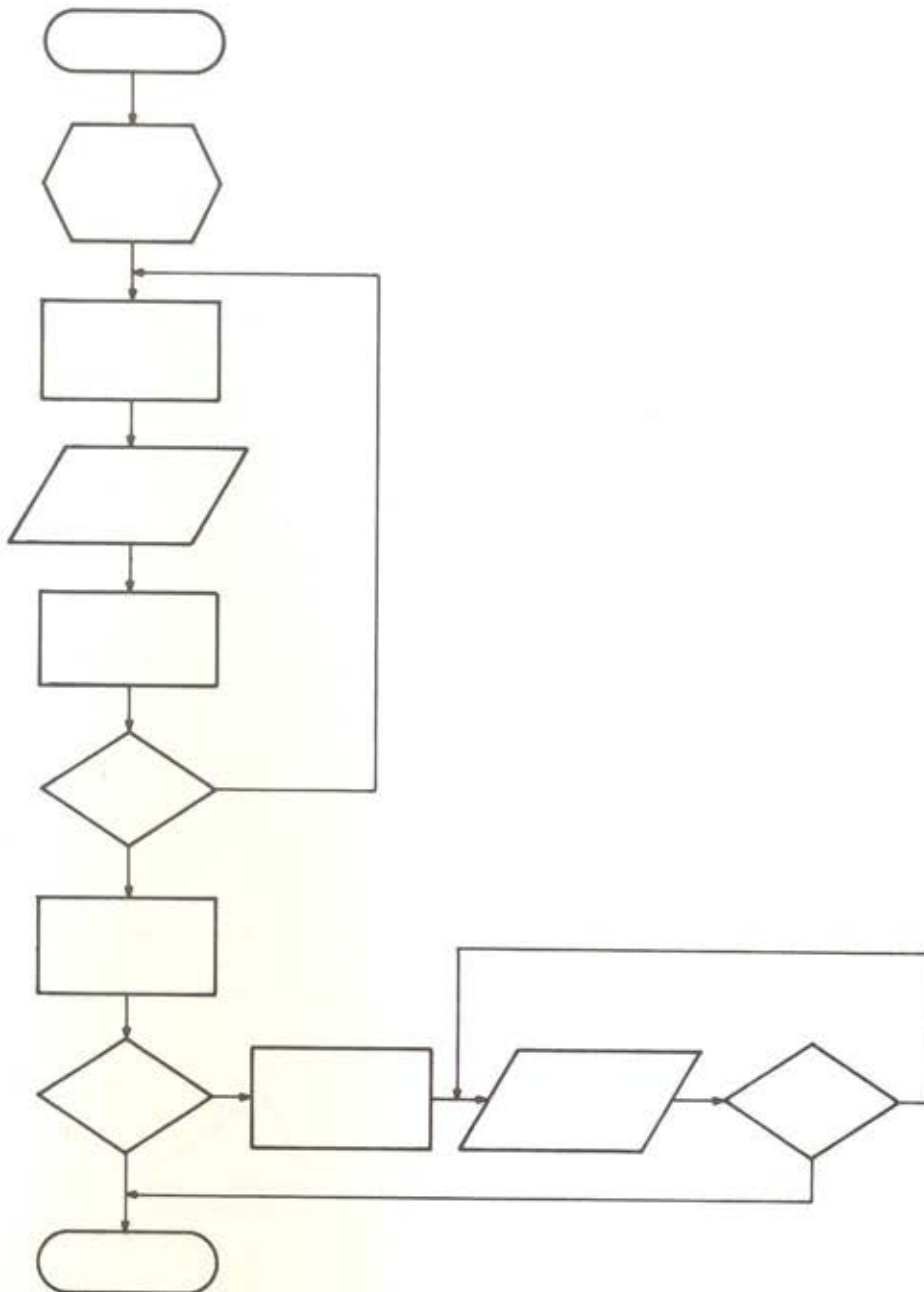
---

### 3. Names and uses of flowchart symbols

- a.  The *process rectangle*. It symbolizes steps that process data.
- b.  The *decision diamond*. It indicates that a choice must be made as to what to do next.
- c.  The *arrow*. It indicates the sequence of the steps.
- d.  The *I/O parallelogram*. It indicates both input and output operations.
- e.  The *initialization hexagon*. It represents setting up beginning conditions.
- f.  The *terminal symbol*. It marks both the beginning and end of the algorithm.
- g.  The *connector*. It is used to make complex flowcharts easier to understand.
- h.  The *predefined process symbol*. It represents a series of steps that appear elsewhere in a flowchart.

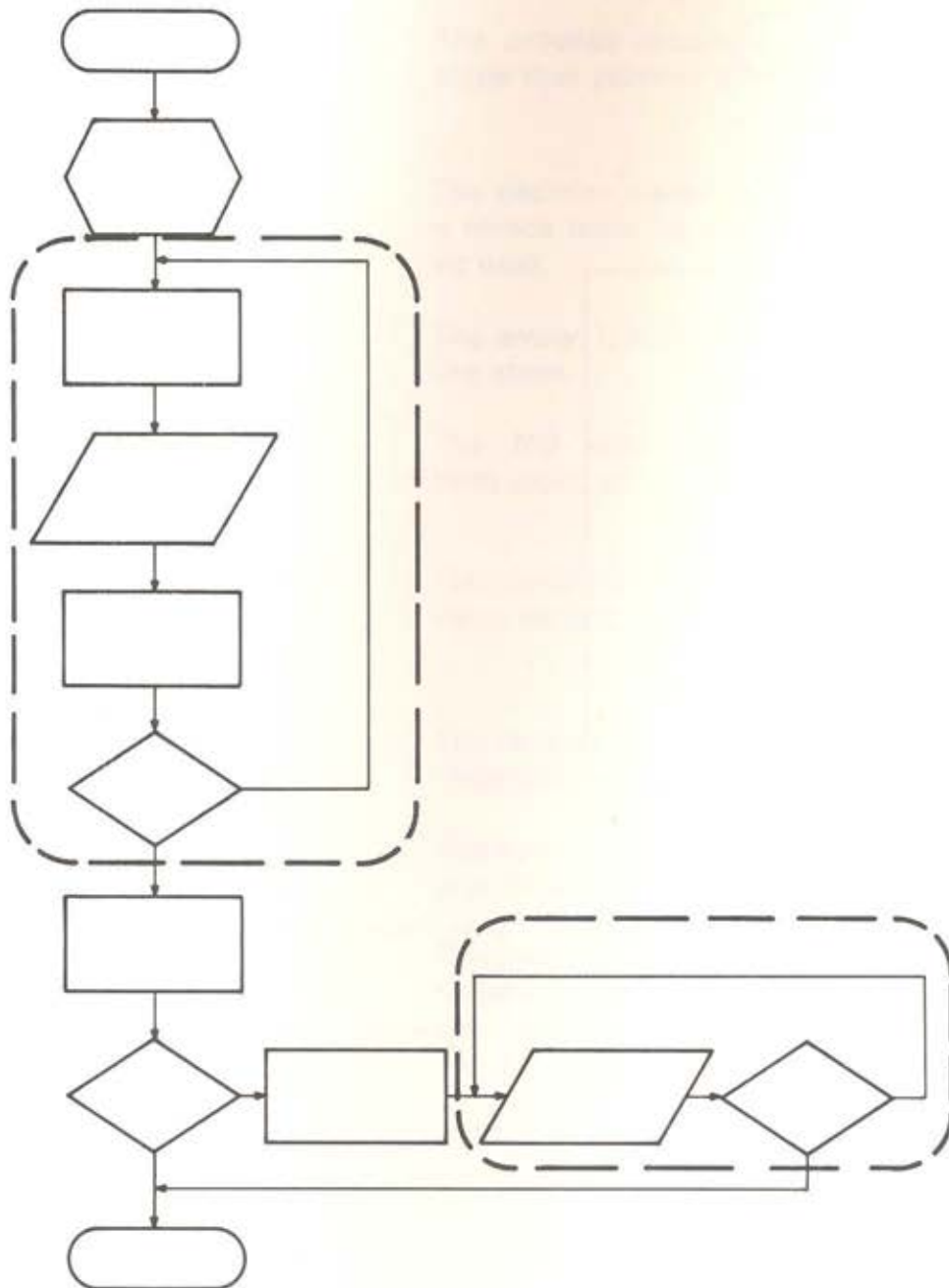
## EXERCISES

4. Circle the loops in the following flowchart.



## SOLUTIONS

4. The loops are circled with dotted lines.

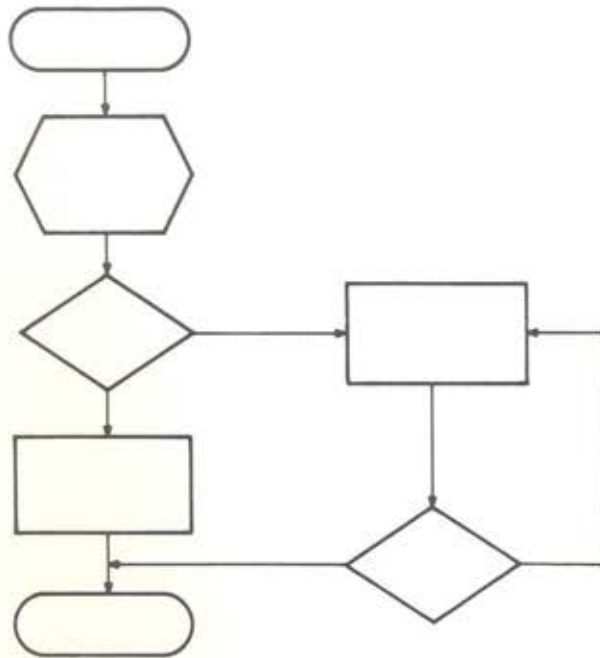


---

## EXERCISES

---

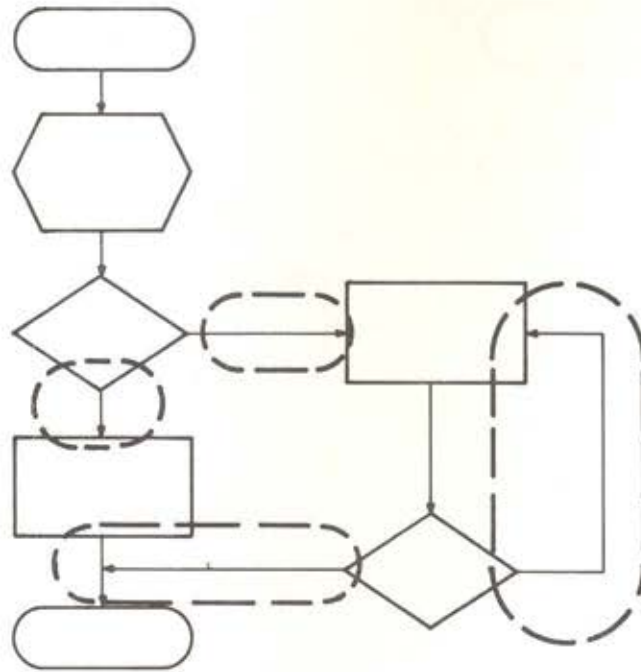
5. Circle all the branches in the following flowchart.



---

**SOLUTIONS**

5. The branches are circled with dotted lines.



## Flowcharting

### OBJECTIVES

1. Given a simple problem, flowchart a solution to the problem using the symbols described in the previous lesson.
2. Given the terms "iteration," "decrement a counter," and "increment a counter" and four definitions for each term, be able to select the correct definition for each term.

### SAMPLE TEST ITEMS

1. Flowchart the solution to the following problem, using the symbols described in the previous lesson.

*Input two unequal numbers and output the higher one.*

2. Circle the letter of the definition of each of these terms:

#### **Iteration**

- a. A single execution of any one instruction in a loop.
- b. A single execution of all instructions in a loop.
- c. A repeated execution of any one instruction in a loop.
- d. A repeated execution of all instructions in a loop.
- e. None of the above.

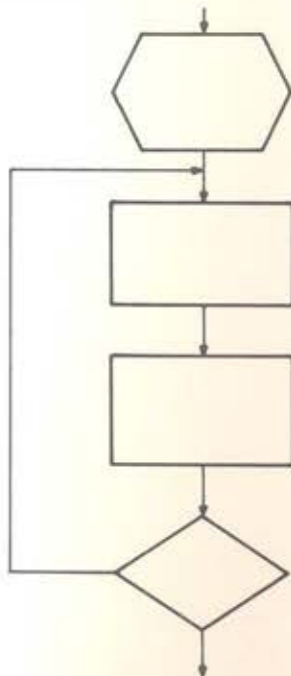
#### **Decrement a Counter**

- a. Subtract some quantity from the number in the counter.
- b. Subtract one from the number in the counter.
- c. Verify the accuracy of the number in the counter.
- d. Check to determine if a loop has been executed the desired number of times.
- e. None of the above.

Mark your place in the workbook and view Lesson 2 in the audio-visual presentation, "Problem Solving."

Now that we know what a flowchart is and how it is used, let's see how we can use flowcharting techniques in more complicated situations. For example, controlled repetition of a series of steps is called *looping*. It is used where many decisions must be made or where step sequences must be repeated a given number of times.

The use of loops is an important flowcharting technique. Loops can be very long and complex. For example, you may have loops *within* loops. Generally, however, you would have this kind of pattern:



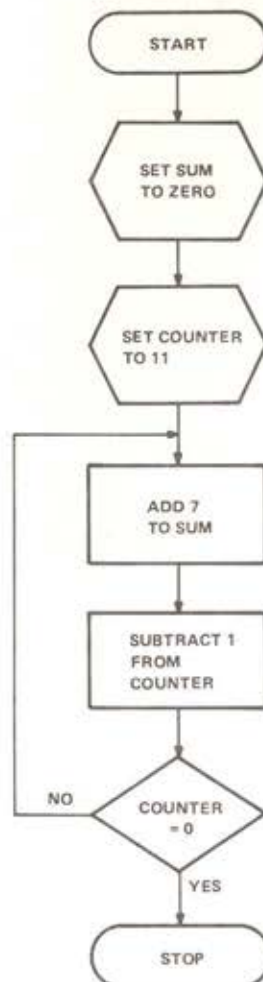
*Initialization* – not really part of the loop, but necessary to set up the beginning condition for the loop.

*A process* you want to execute repeatedly – the reason for constructing the loop.

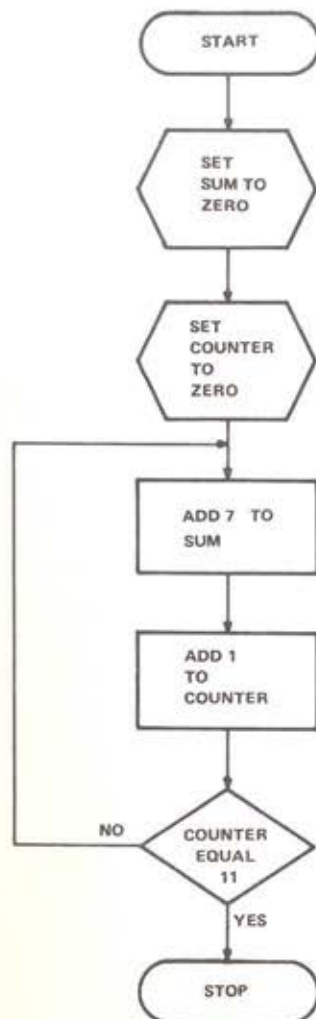
*Subtracting from (or adding to) a counter* – to keep track of how many times the loop is executed.

*Checking* – to find out if the loop has been executed the desired number of times. If not, do it again.

A *single execution* of the instructions in a loop is called an *iteration*. A counter may be used to keep track of the number of iterations. In the audio-visual program, the counter was initialized to 11 and 1 was subtracted from it for each iteration. Subtracting from a counter each time you complete an iteration is called *decrementing* a counter.



However, we could have multiplied  $7 \times 11$  this way:



That is, add 1 to the counter each time you complete an iteration and stop when you reach 11. Adding to a counter is called *incrementing* a counter.

Some points to remember:

1. *No single correct solution* can be found to a flowcharting problem. Even a simple flowchart can usually be drawn several ways, each equally correct. The test of a flowchart is its ability to do the job.
2. *Quality*, of course, varies. That is, some flowcharts are better than others – more efficient (do the job in fewer steps) or clearer (easier to read).
3. *Clarity* is a critical factor in flowcharting. It is hard to appreciate this from the relatively simple flowcharts that have been presented here, but charting a complex problem can result in an incomprehensible mess. A flowchart is a form of communication: if it can't be understood, it is worthless. You have already seen two aids to clarity: the connector and the pre-defined process symbol.

Another way to aid clarity is to adopt a set of standard practices. For example, you might restrict the directions of arrows so that they always point either down or to the right. Exercise 5 is an example of this.

---

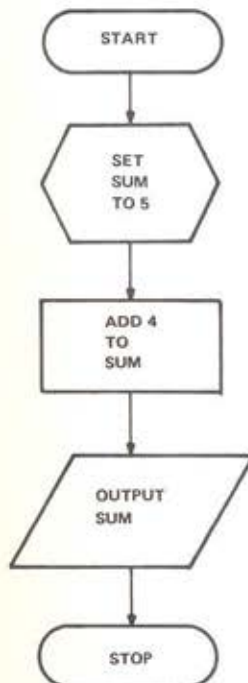
## EXERCISES

---

1. Define the following:

- a. Iteration
- b. Decrement a counter
- c. Increment a counter

2. What will the algorithm represented by this flowchart do?



---

## SOLUTIONS

---

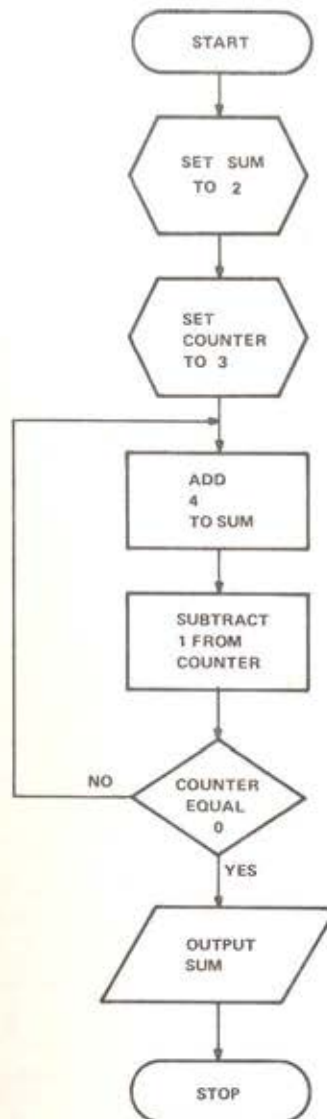
### 1. Definitions

- a. **Iteration** – A single execution of all the instructions in a loop.
- b. **Decrement a counter** – Subtract some quantity from the number in the counter.
- c. **Increment a counter** – Add some quantity to the number in the counter.

### 2. Output a 9.

## EXERCISES

3. What will the algorithm represented by this flowchart do?



---

## SOLUTIONS

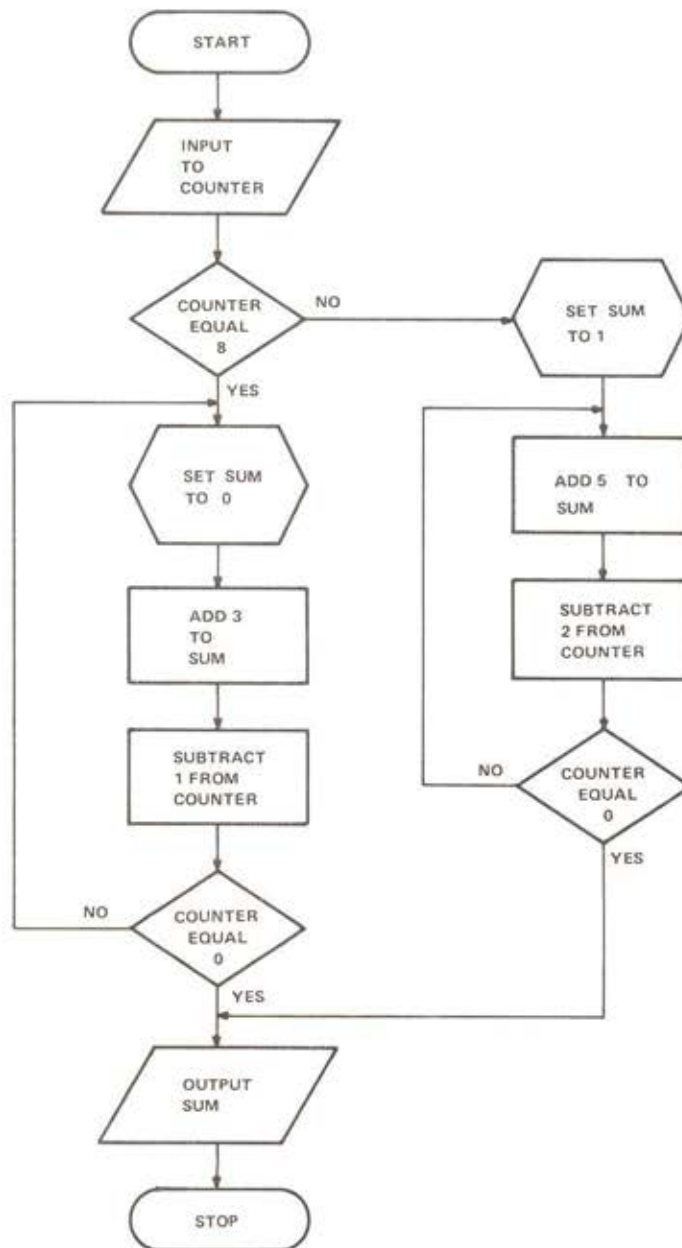
---

3. Output a 14 ( $2 + 4 + 4 + 4$ ). *Sum* was initialized to 2.

## EXERCISES

4. What will result from this algorithm if . . .

- a. 6 is input?
- b. 8 is input?



---

## SOLUTIONS

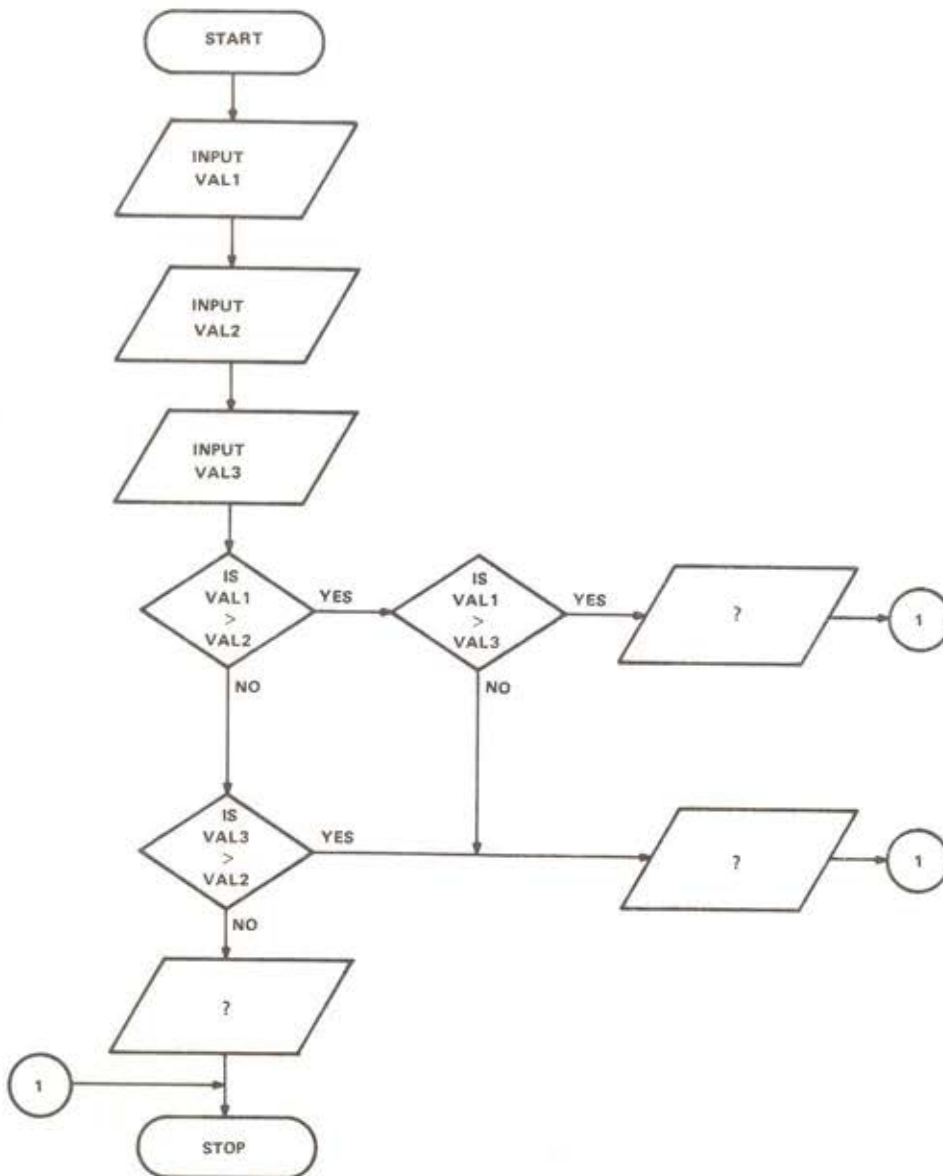
---

### 4. Result of Algorithm

- a. Output 16 ( $1 + 5 + 5 + 5$ ). *Sum* was initialized to 1. Since 2 was subtracted from the counter with each iteration, the loop was only executed three times.
- b. Output 3. Every time the loop was executed, it started by setting *Sum* to 0. Only the last addition was not wiped out.

## EXERCISES

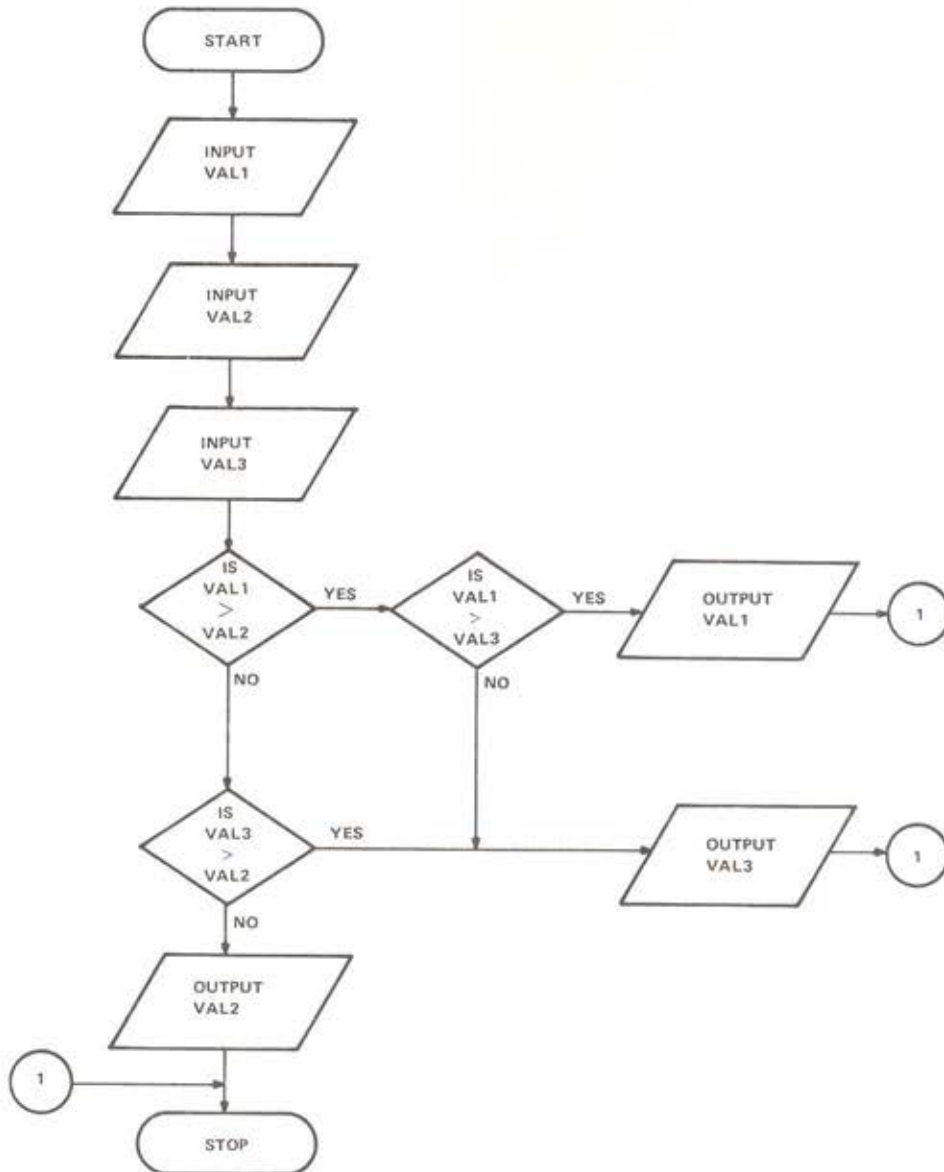
5. Complete this flowchart in order to output the highest of three unequal values.



Note: The symbol ">" means "greater than"; the symbol "<" means "less than."

## SOLUTIONS

5. Answers are indicated in flowchart.



---

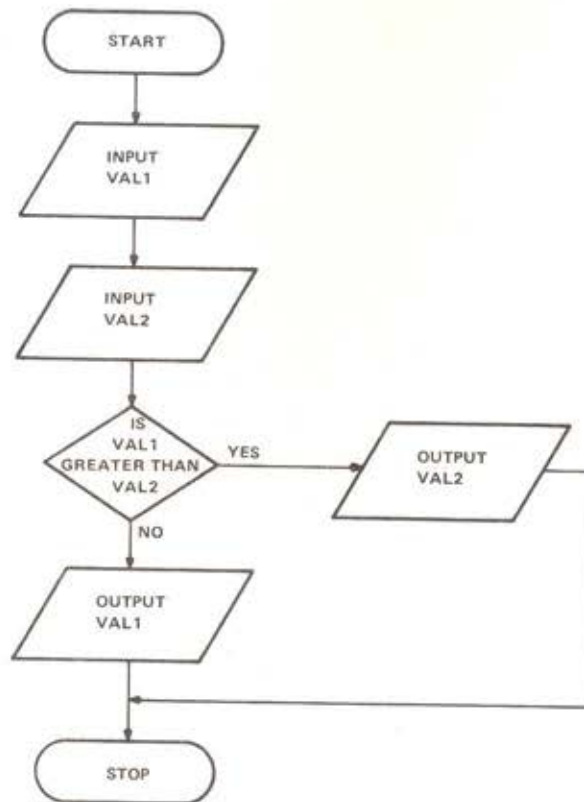
## EXERCISES

---

6. Draw a flowchart that inputs two unequal numbers and outputs the *lower one*.

## SOLUTIONS

### 6. Sample Flowchart



## Steps in Problem Solving

### OBJECTIVE

Given a random ordering of the seven steps to problem solving with a computer and a list of descriptions, be able to: (1) place the seven steps in their chronological order, and (2) match each step with its description.

### SAMPLE TEST ITEMS

The seven steps to problem solving with a computer and their descriptions are given below. Both steps and descriptions are in random order. Your task includes two parts. First, indicate each step's position in chronological order by placing the appropriate number (1 for first step, 2 for second step, etc.) in the column labeled Position. Second, write the letter that correctly describes each step in the column marked Description.

Step	Position	Description
Code the Program	_____	_____
Flowchart	_____	_____
Document	_____	_____
Define the Problem	_____	_____
Debug	_____	_____
Translate	_____	_____
Plan the Solution	_____	_____

#### Descriptions

- a. Determine what needs to be done, what resources are available, and how they will be used.
- b. Write the program in a language the computer can understand.
- c. Describe, in detail, what you want to do and the data you have to work with.
- d. Find and correct errors in processing.
- e. Transform the coded program to a form the computer can execute.
- f. Collect and keep a record of all the information about the program.
- g. Using symbols, draw a detailed diagram of the sequence of steps to reach the desired solution.

Problem solving with the computer centers around the preparation of a sequence of instructions, called a *program*, that tells the computer what to do. This process can be broken down into seven steps:

1. Define the problem
2. Plan the solution
3. Flowchart
4. Code the program
5. Translate
6. Debug
7. Document

### **Define the Problem**

Defining a problem is mainly a process of analysis; that is, breaking it down into elements:

- What you want to do (*output definition*)
- The data you have to work with (*input definition*)

Problems are usually presented in terms of some desired result or output. For example, here is a simple word problem:

If pencils are bought for fifty cents a dozen and sold for sixty cents a dozen, what is the total profit on seven dozen pencils? On twelve dozen? On forty-two dozen?

The output desired is a number – the total profit from the sale of so many dozen pencils.

Now that we know what we want, let's determine what we have to work with. In this case, input consists of information about the price of pencils: they are bought for fifty cents a dozen and sold for sixty cents a dozen.

### Plan the Solution

Now that the problem has been defined, the next step is to plan the solution. There are three aspects:

1. *What needs to be done?*
2. *What resources are available?*
3. *How will they be used?*

In our word problem, we need to perform some calculations on the data about pencils to determine the profit. The available resource is a computer that can add and subtract, but not multiply.

The computer will be used to perform the calculations.

First, to find the profit for one dozen, we subtract the buying price from the selling price:

$$60¢ - 50¢ = 10¢$$

Then, we multiply that figure by the number of pencils (in dozens) sold. For example, if seven dozen are sold:

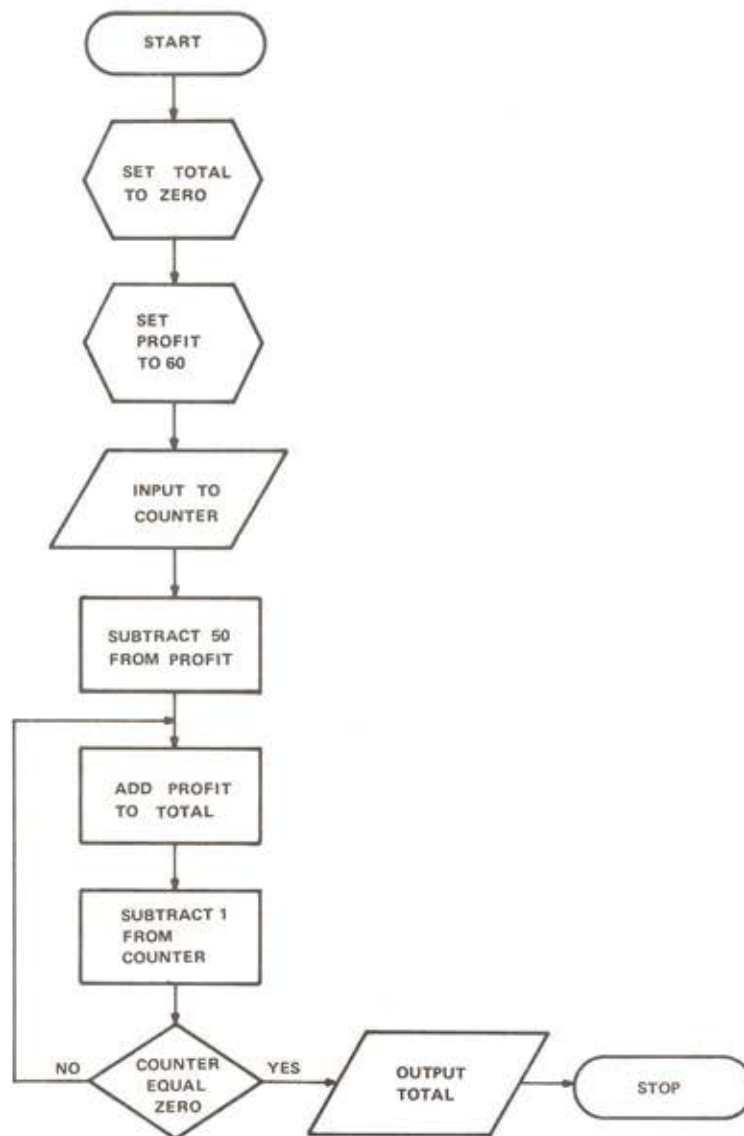
$$10¢ \text{ times } 7 = 70¢$$

The total profit is 70¢.

## Flowchart

The next step involves drawing the detailed plan of how the job will be done. In other words, you flowchart the task the computer will follow to find the solution to this problem.

The flowchart for the word problem might look like this (the number of dozen sold is the input):



## Code the Program

Unfortunately, the computer can't read flowcharts. Therefore, we must use a language the computer can understand. This is called program *coding*. The program can then be input to the computer from a terminal or other input unit.

## Translate

However, yet another step is required before the program can be run. Even though the computer can now read the program, it still can't *execute* the instructions. This requires that the program be in a totally different form. The transformation of a coded program to a form the machine can actually execute is called *translation*. Fortunately, the programmer doesn't have to do this. It is done by the computer, under the control of a program furnished for that purpose.

## Debug

Suppose, after going through all this, the program doesn't work. The output might be wrong or there might be no output at all, as a result of errors made along the way. Such errors are called *bugs*, and the process of finding and correcting them is called *debugging*.

The main point here is to check yourself as you go along. Make certain that:

- The output definition is clear, and it is really what is desired.
- The input information is complete and accurate.
- The procedures depicted in the flowcharts will actually do the job.
- The coded program is correct and accurately reflects the flowchart.

Before a program is used for execution, it should be thoroughly tested. This is generally done by using *test data*. Test data consists of input which resembles the input that will actually be used during execution. The program is run with the test data to make sure it does exactly what it's supposed to do. For example, we might try the pencil problem with, say, two or three dozen and compare the output to answers we calculate manually.

## Document

This brings us to the final step – documentation. Documentation is the collection of all that *anyone* might need to know about the program. It should at least include:

- *Problem Definition* – input and output
- Flowcharts
- Description of how the program was tested – test data and test output
- Name of programmer
- Date of completion

Documentation becomes quite important when somebody wants to change the program, something that is quite likely to happen to a program that is run repeatedly over a long period.

Now do the following exercises before going on to the module test.

---

### EXERCISES

---

- d. Document

---

## SOLUTIONS

---

1. Describe, in detail, what you want to do (the output) and the data you have to work with (the input).
2. Determine what needs to be done, what resources you have to work with, and how you will use them.
3. Using flowchart symbols, draw a detailed diagram of the sequence of steps to reach the desired solution.
4. Definitions
  - a. **Code** – Write the program in a language the computer can understand.
  - b. **Translate** – Transform the coded program to a form the computer can execute.
  - c. **Debug** – Find and correct errors in processing.
  - d. **Document** – Collect and keep a record of all the information about the program.

Take the test for this module and evaluate your answers before studying another module.