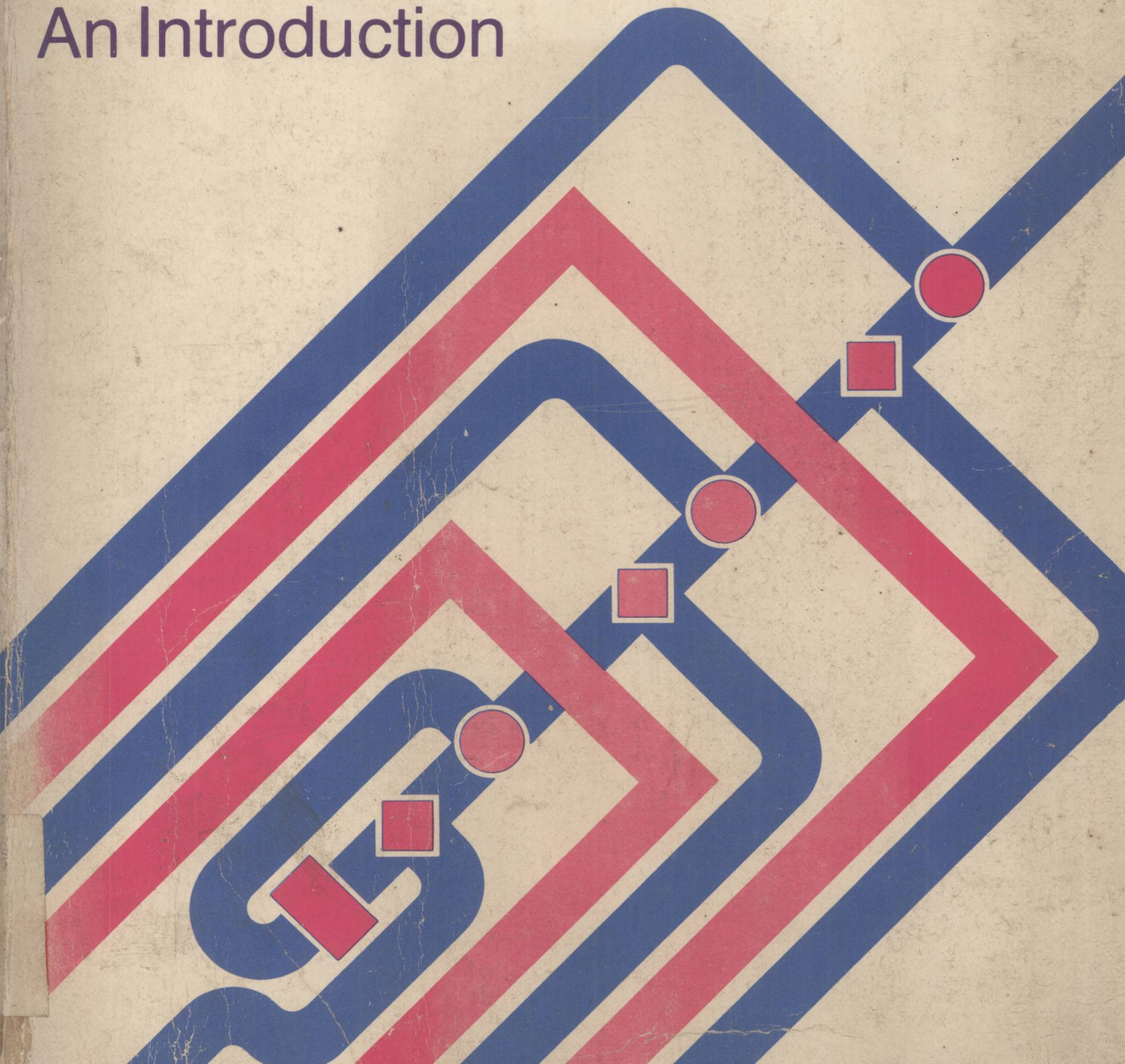


MICHAEL MARCOTTY

Structured Programming with PL/I

An Introduction



TP.321
M321

7960021
5

Structured Programming with PL/I

An Introduction

MICHAEL MARCOTTY

Research Laboratories
General Motors Corporation



E7950021

PRENTICE-HALL, INC., Englewood Cliffs, New Jersey 07632

TP
5

Library of Congress Cataloging in Publication Data

MARCOTTY, MICHAEL (date)

Structured programming with PL/I.

Includes index.

1. PL/I (Computer program language) 2. Structured programming. I. Title.

QA76.73.P25M37 001.6'424 76-26879

ISBN 0-13-854885-4

© 1977 by Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632

All rights reserved. No part of this book may be reproduced in any form or by any means without permission in writing from the publisher.

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

Prentice-Hall International, Inc., *London*

Prentice-Hall of Australia Pty. Limited, *Sydney*

Prentice-Hall of Canada, Ltd., *Toronto*

Prentice-Hall of India Private Limited, *New Delhi*

Prentice-Hall of Japan, Inc., *Tokyo*

Prentice-Hall of Southeast Asia Pte. Ltd., *Singapore*

Whitehall Books Limited, *Wellington, New Zealand*

*Never take anything as true without evidence;
that is, neither leap to conclusions
nor accept something
for which there is the least doubt.*

*Divide each difficulty into as many parts as possible
so that it may be overcome more easily.*

*Starting with the simplest and easiest to understand,
consider things in order, moving by degrees
to the most complex.*

*Finally, make all enumerations so complete
and all reviews so general
that nothing is left out.*

Descartes
Discourse on Method (1637)

Preface

This book is an introduction to computer programming that does not require any advanced mathematics. Many people think they could never program a computer because they are frightened by a square root symbol. Most computer programming does not require a knowledge of mathematics beyond simple arithmetic; it is much more important to be able to think clearly and logically. This book is mainly addressed to people like this. However, for those who enjoy slightly more mathematical fare, there are sections (marked with asterisks) where some mathematical ability is assumed.

Although the emphasis here is on programming rather than on a programming language, we need a language to describe programming. The language PL/I is used here because, of the generally available languages, PL/I is the best for explaining the programming process. Learning the first language is the hardest; once the principles of programming have been learned, a second language comes easily.

There is no attempt to cover the whole of PL/I; no tutorial text that I know of has attempted that task! Instead, I have taken the opposite direction. My aim is to introduce the minimum of PL/I that will allow me to explain the basics of programming. By taking this approach, I have been able to remain well within almost all implementations of PL/I and its subsets. In particular, all the programs described can be run with the PL/C subset, which is becoming more and more popular.

One of the problems with PL/I is its size, brought about by a multiplicity of options and their consequent rules. I have reduced this complexity in two ways: by describing only the small part of the language that I need for my explanation of programming and by giving rules that are more restrictive than they need be—

according to PL/I. For example, I do not discuss the GO TO statement, and I insist that all variables be declared. The first because, at least at the level of this book, the use of the GO TO statement offers no advantages and many disadvantages. The second because the clarity of all programs is improved by declaring the variables. Those of you who already know PL/I may well find that I have omitted your favorite feature or restricted one of your pet forms out of existence. So be it. I have chosen a subset that will allow me to solve a wide range of problems yet keep the number of rules to be explained to a minimum.

In programming, correctness is everything. To help achieve this, the top-down method of program development is used throughout. At each stage, the partly complete program is reviewed for correctness. The key to bugfree programs is to avoid the errors in the first place. I try to show by example a style of program development that will help the student write correct programs. The programmer must learn to understand clearly what each statement in a program does. Far too many programs are constructed from partly understood statements combined without thought of structure and tested with haphazardly chosen data. It is for this reason that, for example, a considerable amount of time is spent discussing the precision of fixed-point arithmetic operations. An understanding of the arithmetic that is used to solve a problem is essential to *knowing* rather than *thinking* that the program is correct. There is nothing deeply mathematical about these rules—it is really a question of bringing back into consciousness the rules of arithmetic that we learned in grade school.

All arithmetic variables and constants in this book, except in the sections marked with an asterisk, are fixed-point decimal. This allows the students to work with the kind of numbers they have been using since grade school. The fact that the computer may store numbers in other than decimal forms is of little concern in learning the basics of programming. The understanding of the algorithm is far more important.

The chapters should be read sequentially. Each one assumes the contents of its predecessors. The sections marked with an asterisk can be omitted without loss to the basic course. For reference purposes, there is an appendix that provides a summary of all the features of PL/I described in this book.

I am very grateful for assistance from many sources. First, to G. Patrick Graham of Wayne State University, who suggested that some notes that I had developed for an introductory programming course could be developed into a book; to the Applied Management and Technology Center of Wayne State University for supplying me with classes and computer time to develop the notes. I am also grateful to the students of those classes for their help with the notes, to Joan K. Hughes for her thorough review of the first draft of the manuscript, and to Sister Ignatia Frye, IHM, of Marygrove College for her assistance in preparing the index. Special thanks go to Eleanor Hiatt of Prentice-Hall for making my introduction to book production such a pleasurable experience. I also thank Donald E. Hart, Head, and George G. Dodd, Assistant Head, of the Computer Science Department of General Motors Research Laboratories for their support. Finally, I must thank my family for enduring my periods at the typewriter so cheerfully.

Contents

Preface *xi*

1 What Is Data Processing? *1*

- 1-1 A Simple Banking Example *1*
- 1-2 Using a Flowchart *5*
- 1-3 Computers *9*
- Summary *21*
- Exercises *21*

2 Introduction to a Programming Language *24*

- 2-1 A Programming Language *24*
- 2-2 Nine PL/I Statements *26*
- 2-3 A PL/I Program *28*
- 2-4 Executing the Program *36*
- 2-5 Some Basic Rules of Programming PL/I *53*
- 2-6 Names in PL/I *54*
- 2-7 The PROCEDURE Statement *55*
- 2-8 The DECLARE Statement *56*
- 2-9 The DO WHILE Statement *57*
- 2-10 The Assignment Statement *58*
- 2-11 Input and Output Statements *59*
- Summary *59*
- Exercises *60*

3	Elementary Data Processing Examples	63
3-1	Running Totals: The Need for Initial Values	63
3-2	Customer Billing: A More Complicated Calculation	69
3-3	Analysis of Grades: Finding Averages	75
3-4	Cost Calculation: Initializing Variables from Input Data	80
*3-5	Floating-Point Variables	88
*3-6	Right-angled Triangles: Powers	93
	Summary	94
	Exercises	96
4	The Development and Check-out of a Program	99
4-1	The Problem Statement	100
4-2	The Development of the Method	101
4-3	Compiling the Program	105
4-4	The Second Compilation	110
4-5	Checking Out the Program	113
	Summary	119
	Exercises	120
5	More Data Processing Problems	124
5-1	Sales and Commission Report: A Loop Within a Loop	124
5-2	Another Commission Problem: Making Decisions	131
5-3	A Classification Problem: More Complicated Decisions	137
5-4	Some Helpful Shortcuts: Built-in Functions	142
*5-5	Some Mathematical Built-in Functions	145
	Summary	147
	Exercises	148
6	Introduction to Software Systems	151
6-1	Operating Systems	152
6-2	Multiprogramming	154
6-3	Executing the Program	157
6-4	Interactive Use of a Computer System	158
6-5	Internal Representation of Programs and Data	160
6-6	Machine Instructions	163
6-7	Programming Languages	165
	Summary	170
	Exercises	171

7 Programming Analysis and Preparation 172

- 7-1 The Problem Statement 172
- 7-2 Algorithms 174
- 7-3 Developing the Algorithm 176
- 7-4 An Alternative to Flowcharts 182
- 7-5 Advantages and Disadvantages of Flowcharts 183
- 7-6 Parallel Example 186
- Summary 190
- Exercises 191

8 Printing Titles in the Output 194

- 8-1 The PUT LIST Statement 195
- 8-2 Character-String Constants 199
- 8-3 Character-String Variables 202
- 8-4 Concatenation and Variable Length Character-String Variables 205
- 8-5 The GET LIST Statement 208
- 8-6 The Conversion of Arithmetic Values to Character-Strings 210
- 8-7 The Null Character-String and the LENGTH Built-in Function 213
- 8-8 The Built-in Functions INDEX and SUBSTR 215
- Summary 220
- Exercises 221

9 Variables and Values 224

- 9-1 Constants and Variables 224
- 9-2 Declaring Character-String Variables 227
- 9-3 The Precision of Arithmetic Variables 230
- 9-4 The Precision of Arithmetic Constants 232
- 9-5 The Assignment of Arithmetic Values 232
- 9-6 Arithmetic Expressions 234
- 9-7 The Precision of the Result of Addition and Subtraction 236
- 9-8 The Precision of Multiplication 241
- 9-9 The Precision of Division 242
- 9-10 An Example 243
- *9-11 The Precision of Floating-Point Numbers 249
- Summary 250
- Exercises 251

10 More About Sequence of Control 254

- 10-1 The Loop 254
- 10-2 Relations and Flags 260
- 10-3 The ENDFILE Condition 263
- 10-4 The Not Operator 265
- 10-5 Compound Relations 266
- 10-6 The Use of the Truth Table 271
- 10-7 Errors with Loops 273
- 10-8 Another Form of the DO Statement 277
- Summary 283
- Exercises 284

11 Lists of Values 286

- 11-1 An Introductory Example 286
- 11-2 Simple Lists 288
- 11-3 Other Properties of Lists 293
- 11-4 Sorting a List of Names 298
- 11-5 Arrays 303
- Summary 307
- Exercises 307

12 Data Structures 310

- 12-1 Working with Simple Structures 310
- 12-2 Reading and Writing Records 320
- 12-3 More About Structuring Data 323
- 12-4 Lists of Structures 324
- 12-5 Searching for an Entry in a List 329
- Summary 336
- Exercises 337

13 More About Input and Output 340

- 13-1 Files 340
- 13-2 OPEN and CLOSE Statements 344
- 13-3 GET and PUT Statements 346
- 13-4 Edit-directed Input and Output 348
- DATA-FORMAT ITEM: A(w) or A 351

	DATA-FORMAT ITEM: F(w) and F(w, d)	351
	CONTROL-FORMAT ITEM: X(w)	352
	CONTROL-FORMAT ITEM: SKIP(n)	353
	CONTROL-FORMAT ITEM: LINE(n)	354
	CONTROL-FORMAT ITEM: PAGE	354
	CONTROL-FORMAT ITEM: COLUMN(n)	354
*13-5	The E Format	354
	DATA-FORMAT ITEM: E(w, d)	355
13-6	Printing a File	355
	Summary	364
	Exercises	364
14	Independent Subprocedures	367
14-1	External Procedures	368
14-2	External Variables	372
14-3	Arguments and Parameters	373
14-4	Expressions and Constants as Arguments	385
14-5	Function Procedures	388
	Summary	390
	Exercises	391
15	Program Development and Verification	392
15-1	The Inventory Update Problem	392
15-2	Solution: Top Level	395
15-3	Completing the Main Procedure	396
15-4	Designing the Test Data	403
15-5	The First Computer Run	407
15-6	Further Development of the Program	409
	Summary	411
	Exercises	411
Appendix I	Glossary	413
Appendix II	Summary of PL/I Features Described in This Book	422
Appendix III	List of PL/C Reserved Words	439
Index		441

1

What Is Data Processing?

A difficulty with beginning to study computer programming is appreciating what it is all about. What is a program? What does it do? What is the connection between a program and a computer? In this chapter we will try to get some feeling for these ideas by discussing a very simple example of data processing in an almost microscopic bank. We will then describe the components of a typical computer system and see how they relate to our banking example.

1-1 A Simple Banking Example

We will start by looking at a very simple banking example. The Norland Bank is a small bank that has very little business—so small, in fact, that one man, Mr. Marshall, looks after everything, and no client has more than one check or one deposit on any day. Because of this low volume of transactions, Mr. Marshall is able to do everything by hand and keep all the balances up to date on a daily basis.

Mrs. Martha Brooks is a client at the Norland Bank. On June 12 Mrs. Brooks has a balance of \$12.00 in her account. She comes into the bank in the morning and makes a deposit to her account of two checks amounting to \$5.00. To make this deposit, she fills in the deposit slip shown in Fig. 1-1 and gives it and the two checks to Mr. Marshall. Martha Brooks' name is printed on the deposit slip so Mr. Marshall can remember who made the deposit. Mr. Marshall keeps the deposit slip as a *record* of the transaction so that he will be able to calculate Mrs.

CHECKING ACCOUNT DEPOSIT TICKET			
Martha Brooks			
DATE <u>June 12</u> 19 <u>XX</u>		CASH →	
CHECKS AND OTHER ITEMS ARE RECEIVED FOR DEPOSIT SUBJECT TO THE TERMS AND CONDITIONS OF THIS BANK'S COLLECTION AGREEMENT.		C	3 70
		H	1 30
		E	
		C	
		K	
		S	
TOTAL FROM OTHER SIDE			
TOTAL ITEMS		TOTAL 5 00	
NORLAND BANK		USE OTHER SIDE FOR ADDITIONAL LISTING ENTER TOTAL HERE BE SURE EACH ITEM IS PROPERLY ENDORSED	

Figure 1-1 Martha Brooks' deposit slip

Brooks' new balance on June 13. He puts the deposit slip on a pile with all the other deposit slips that he receives during that day.

Also on June 12 Mrs. Brooks writes a check for \$9.00, Fig. 1-2, to the Readmore Bookstore just around the corner from the Norland Bank. Later that day Roger Lands, manager of the Readmore Bookstore, brings Mrs. Brooks' check around to Mr. Marshall who gives Mr. Lands the \$9.00. Mr. Marshall keeps the check as a record of that transaction. He knows that it is Mrs. Brooks' check because he recognizes her signature. Although he knows the signature of everyone of his clients, he still insists that their names be printed on the checks as an extra safeguard. Mr. Marshall puts the check in the pile with the other checks that had been brought in during the day.

First thing in the morning of June 13 Mr. Marshall calculates Mrs. Brooks' new account balance, taking into account the two transactions of the previous day. How does he do it?

To begin with, he takes the pile of deposit slips that he received the day before and *sorts* them into alphabetical order (Fig. 1-3). Then he does the same thing for the pile of checks. He now has two piles of input documents that he can

DATE <u>June 12</u> 19 <u>XX</u>		NO. <u>XX</u>
PAY TO THE ORDER OF	<u>Readmore Bookstore</u>	\$ <u>9.00</u>
<u>Nine</u>		DOLLARS
NORLAND BANK		Martha Brooks
MEMO	<u>Martha Brooks</u>	

Figure 1-2 Martha Brooks' check

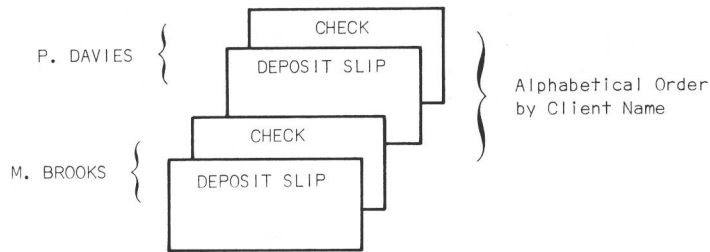


Figure 1-3

merge into a single pile, still in alphabetical order. There are two documents for each client. As he pairs them, he staples them together so that he can *process* the *data* on the two documents easily. Since Mrs. Brooks' name is first alphabetically among the clients at Mr. Marshall's bank, her input records are on the top of the batch after sorting.

Next, Mr. Marshall starts to process the data and prepares a work sheet, Fig. 1-4, on which to do his figuring. On the first line of his work sheet, under the heading CLIENT, he writes *Martha Brooks*. The next thing to be done is to write in the OLD BALANCE column Mrs. Brooks' current bank balance. So, Mr. Marshall reaches for the bank's client *file*, which has a *record* of the current day's balance in each client's account. The records in this file are kept in alphabetical order so that they are in the same order as the input documents. He goes to the file, picks out Mrs. Brooks' record, and finds that the balance in her account is \$12.00. This is the amount Mr. Marshall lists on the first line in the OLD BALANCE column of his worksheet. When Mr. Marshall removes a record from the file, he finds it helpful to mark the place with a block of wood. Because he is working his way steadily through the file, the block of wood also serves to mark how far he has got and saves Mr. Marshall from having to search from the beginning of the file for each record.

CLIENT	OLD BALANCE	DEPOSIT	PAYMENT	NEW BALANCE
Martha Brooks	12.00	5.00	9.00	8.00
Peter Davies	5.00	8.00	4.00	9.00
David A. Dobbs	13.12	24.19	27.82	9.49
James Ellis	568.72	163.71	147.51	584.92
Anne R. Fish	62.87	27.82	16.17	74.52
Roger Hands	2.13	4.17	1.15	5.15
Peter T. Stevens	121.34	51.72	5.23	167.83

Figure 1-4 Mr. Marshall's work sheet

In the DEPOSIT column, on the same line, he lists the data item \$5.00 from the deposit slip and, in the PAYMENT column, the data item \$9.00, which is the amount of Mrs. Brooks' check of June 12. He then takes a scratch pad and does the arithmetic:

$$\begin{array}{r}
 12.00 \\
 + \quad 5.00 \\
 \hline
 17.00 \\
 - \quad 9.00 \\
 \hline
 8.00
 \end{array}$$

and enters \$8.00 in the NEW BALANCE.

Following the same method, Mr. Marshall completes his work sheet, line by line, transaction by transaction, for all the input documents. At the same time, he updates the client records in the client file with the data from the work sheet. When he has finished, Mr. Marshall takes his work sheet and types the Daily Transaction Report from it. This serves as a quick reference to establish recent transactions and the current balance of clients.

Mr. Marshall does not throw away the input documents immediately. Instead, he keeps them for two days to act as a control in case there is some question that he might have made a mistake. They also serve as a kind of insurance or backup should the Daily Transaction Report be lost and the day's data processing has to be repeated.

This concludes our description of the way in which Mr. Marshall keeps his clients' bank accounts up to date. This is the data processing work for the Norland Bank and we can summarize it in ten steps:

1. Sort and merge the deposit slips and checks (the input documents) into alphabetical order by client name, a pair of documents for each client.
2. Retrieve the client's record from the client file, copy the client's name and current balance into the OLD BALANCE column of the work sheet.
3. Copy the amount of the deposit from the deposit slip into the DEPOSIT column of the work sheet.
4. Copy the amount of the check into the PAYMENT column of the work sheet.
5. Save the input documents for control and backup.
6. Calculate the new balance and enter it into the NEW BALANCE column of the work sheet.
7. Update the client's record by entering the new balance from the NEW BALANCE column.

8. Replace the record in the file.
9. Repeat steps 2 through 8 for all input records.
10. Type the completed work sheet as the Daily Transaction Report.

A description of a data processing task as a set of steps to be performed is a *procedure*.

If we look carefully at Mr. Marshall's procedure for processing the data of the Norland Bank, we see that he spends only a small fraction of his time calculating. Most of his time is spent in getting the data organized and ready so that the calculation can be performed. He also spends much time updating records and preparing his work sheet so that he can type the Daily Transaction Report. Calculating is only one of many operations that must be performed in a data processing job.

1-2 Using a Flowchart

Later that summer Mr. Marshall hires a high-school student, Jim Brown, to help him with the data processing task. Since Jim has no experience, Mr. Marshall makes the work as simple as possible, giving Jim just the calculations and typing of the report to do. Mr. Marshall does all the sorting of the input data and searching of the client file himself. He writes the numbers for Jim on a card; one client's data on one card. To keep the confidentiality of the bank's work, instead of writing the client's name on the card, Mr. Marshall gives each client a number and writes it on the card. So that Jim knows which number on the card is which, Mr. Marshall labels the numbers on the cards as shown in Fig. 1-5. Without the labels, it would be too easy for Jim to forget the order of the numbers and make mistakes. Mr. Marshall calls the set of cards the *input card deck*.

Mr. Marshall also prepares some very simple instructions for Jim, telling him exactly what he has to do. To make it easier, Mr. Marshall expresses the instructions in a diagram, shown in Fig. 1-6, called a *flowchart*. This flowchart is a com-

CLIENT NUMBER=7 OLD BALANCE = 121.34 DEPOSIT= 51.72 PAYMENT= 5.23
CLIENT NUMBER=6 OLD BALANCE = 2.13 DEPOSIT= 4.17 PAYMENT= 1.15
CLIENT NUMBER=5 OLD BALANCE = 62.87 DEPOSIT= 27.82 PAYMENT= 16.17
CLIENT NUMBER=4 OLD BALANCE = 568.72 DEPOSIT= 163.71 PAYMENT= 147.51
CLIENT NUMBER=3 OLD BALANCE = 13.12 DEPOSIT= 24.19 PAYMENT= 27.82
CLIENT NUMBER=2 OLD BALANCE = 5.00 DEPOSIT= 8.00 PAYMENT= 4.00
CLIENT NUMBER=1 OLD BALANCE = 12.00 DEPOSIT= 5.00 PAYMENT= 9.00

Figure 1-5 The input card deck

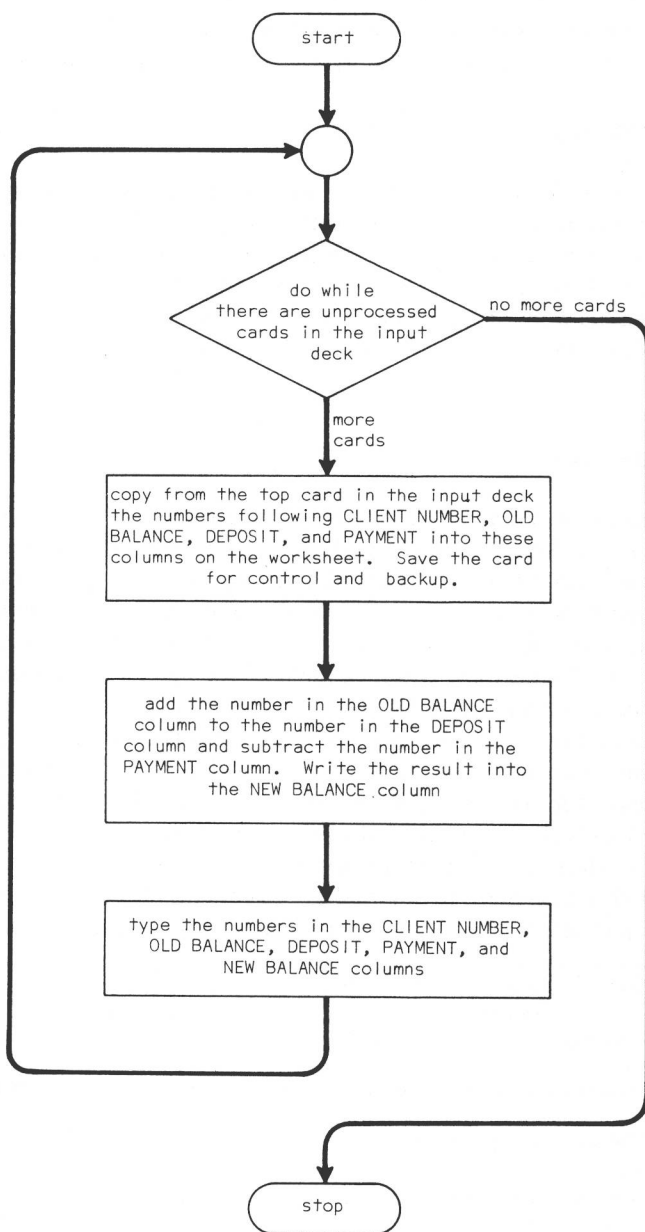


Figure 1-6 Mr. Marshall's flowchart