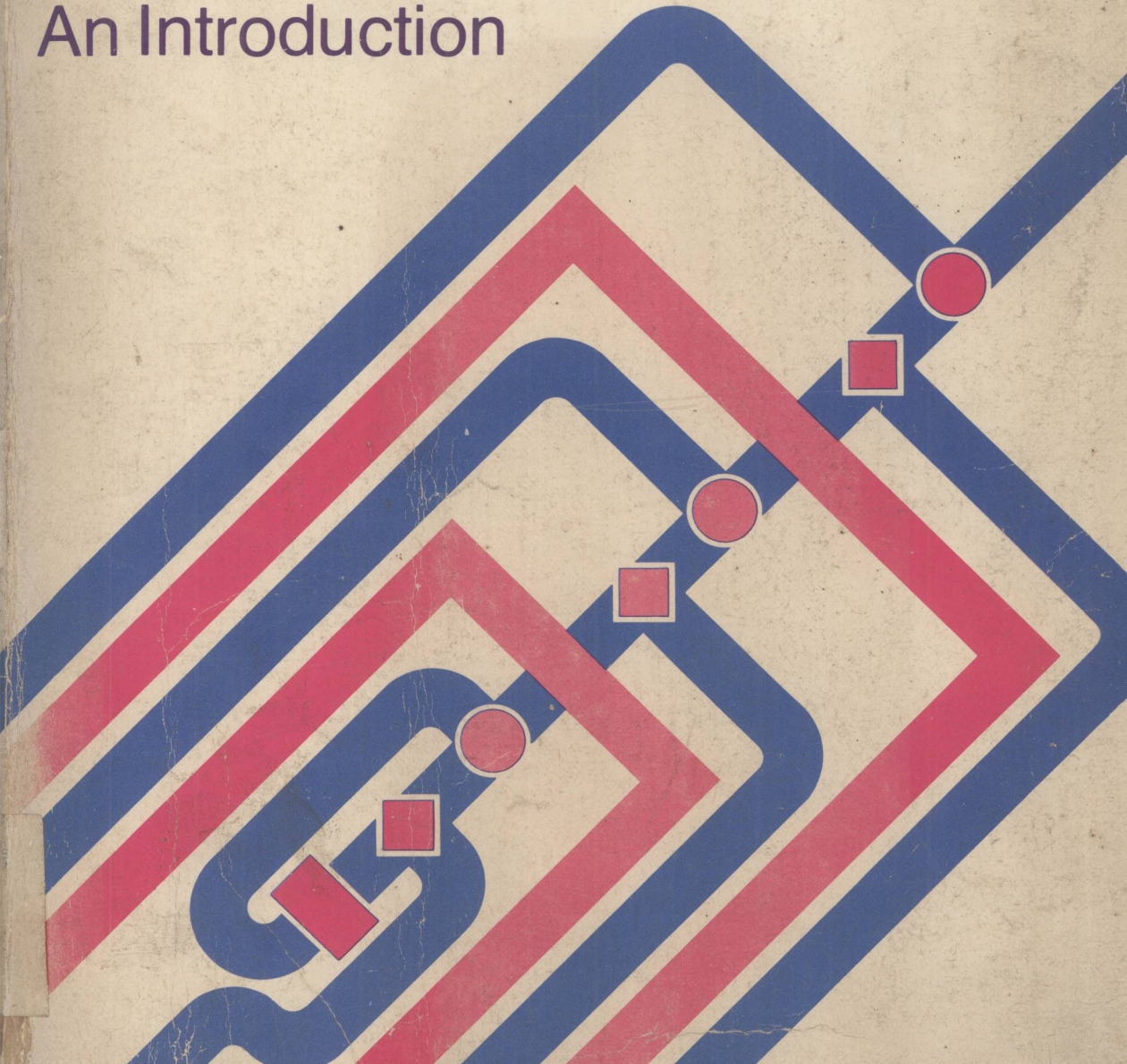
MICHAEL MARCOTTY

Structured Programming with PL/I

An Introduction



Structured Programming with PL/I

An Introduction

MICHAEL MARCOTTY

Research Laboratories
General Motors Corporation

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

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

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 artithmetic 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

1

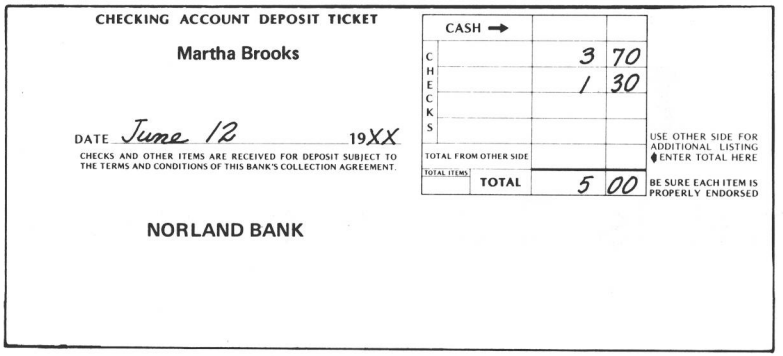
What Is Data Processing?

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 June 12 19XX

CHECKS AND OTHER ITEMS ARE RECEIVED FOR DEPOSIT SUBJECT TO THE TERMS AND CONDITIONS OF THIS BANK'S COLLECTION AGREEMENT.

NORLAND BANK

CASH →			
CHECKS		3	70
		1	30
TOTAL FROM OTHER SIDE			
TOTAL ITEMS	TOTAL	5	00

USE OTHER SIDE FOR ADDITIONAL LISTING
ENTER TOTAL HERE

BE SURE EACH ITEM IS PROPERLY ENDORSED

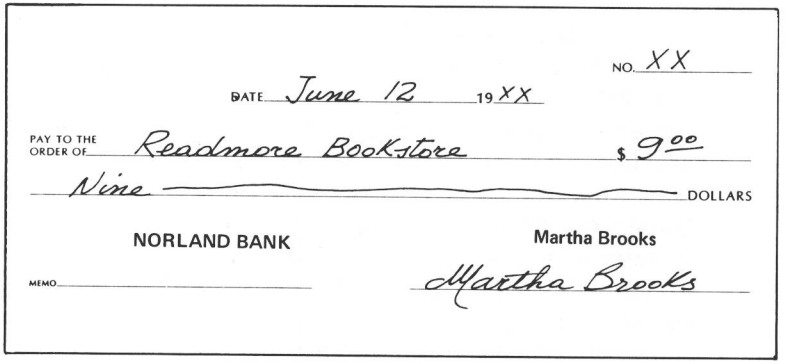
Figure 1-1 Martha Brooks' deposit slip

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


NO. XX

DATE June 12 19XX

PAY TO THE ORDER OF Readmore Bookstore $ 9.00

Nine DOLLARS

NORLAND BANK

Martha Brooks

MEMO

Martha Brooks

Figure 1-2 Martha Brooks' check

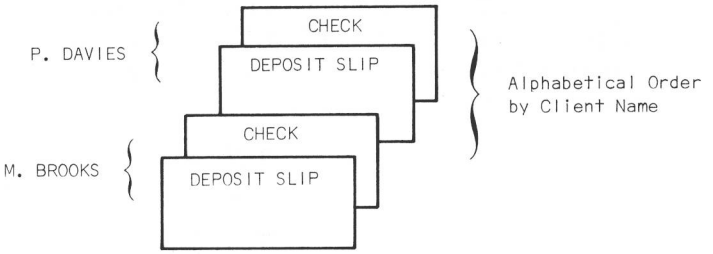


Figure 1-3

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 Lands	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:

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

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 lables, 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-

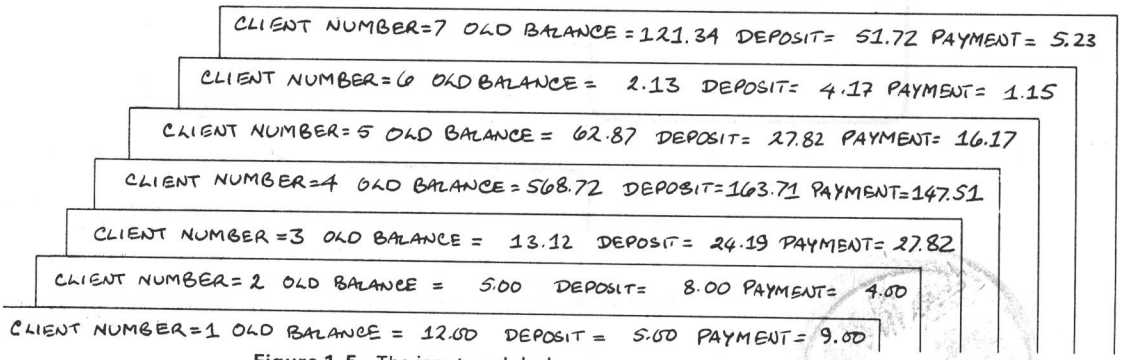


Figure 1-5 The input card deck

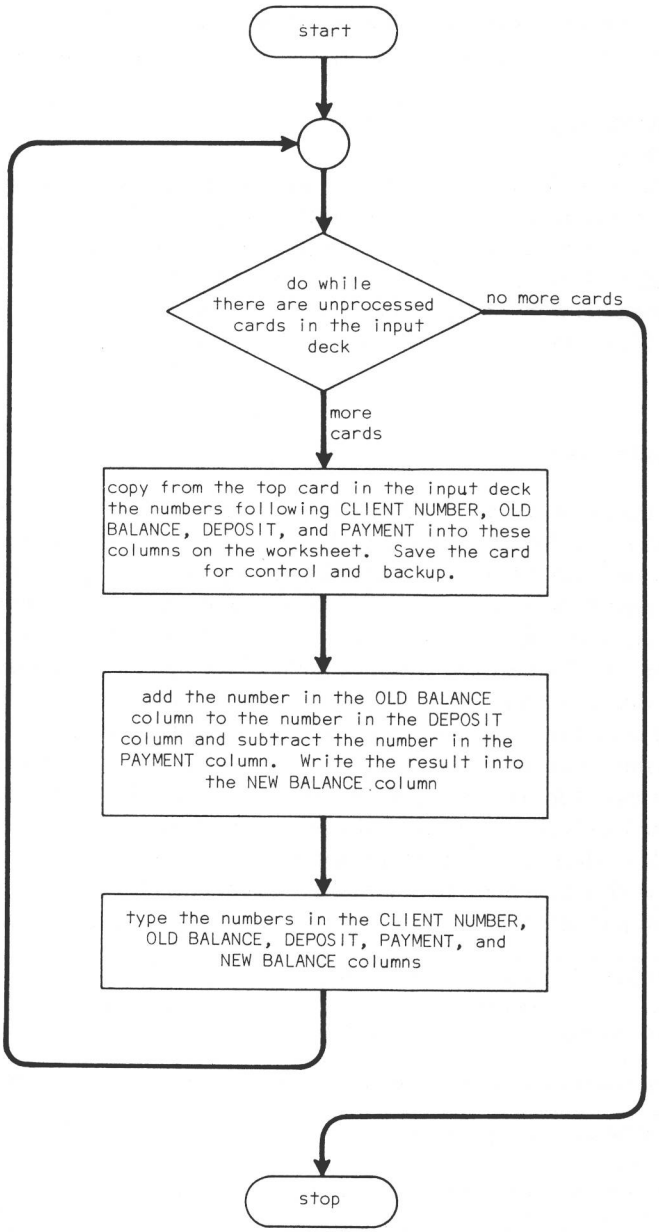


Figure 1-6 Mr. Marshall's flowchart