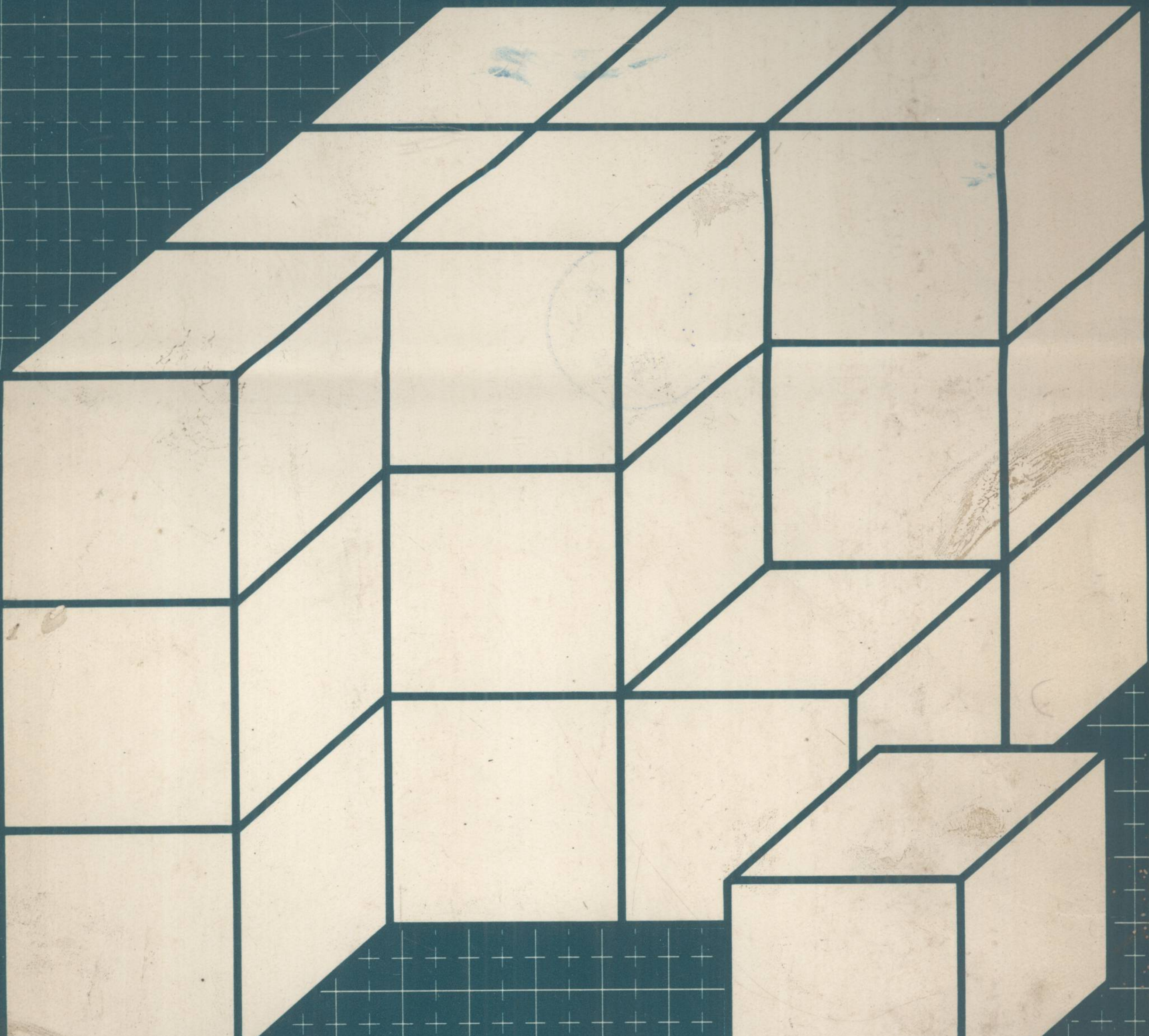
Instructor's Guide

PROGRAMMING LOGIC

Marjorie Leeson



INSTRUCTOR'S GUIDE

PROGRAMMING LOGIC

MARJORIE LEESON
DELTA COLLEGE

SRA®
SCIENCE RESEARCH ASSOCIATES, INC.
Chicago, Palo Alto, Toronto, Henley-on-Thames, Sydney
A Subsidiary of IBM

ISBN 0-574-21421-6

Printed in the United States of America.

10 9 8 7 6 5 4 3 2 1

PREFACE

Since teaching programming logic is not an easy task, SRA and the author would like to do everything possible to assist you. Therefore, this guide contains material suggested by the respondents of a national survey. Teachers who responded to the survey were either teaching a programming logic course or were contemplating doing so. Many of their suggestions are incorporated into the text and into the instructor's guide.

As indicated in the preface of the text, some rather ambitious goals were established--goals that necessitated making assumptions about the background, interests, and commitment of students who will use the text and faculty who will teach from the text.

First, it is assumed that most students have taken a previous course in fundamentals of computer processing. While it is intended to present fully the fundamental tools and techniques of programming, material typically included in a computer concepts course is only briefly reviewed. When the material in the text was field tested, students without an introductory course did as well as the students who had taken such a course. However, students without an introductory course did spend more time learning key words and in answering the checkpoint questions.

Secondly, it is assumed that students have an interest in business. Students who are taking or who have taken business courses feel comfortable with the terminology and with the assignments. Students who have not had business courses usually find the realistic approach used in the text worth the small effort required to learn more about basic business and data processing concepts.

Thirdly, it is assumed that the reader is committed to learning good programming skills. As stated in the text's preface, "those who are not willing to work to develop such skills will soon be left by the wayside." This is not a subject that can be learned without effort. As a classroom teacher, I am sure you agree that students considering programming as a possible career choice must be fully prepared for the self-discipline and analytical skills the field requires. Other students who may not elect to become programmers will develop an appreciation of the skills required to design and develop both programs and systems.

Similar assumptions have been made regarding teachers using this book. First, it is assumed that instructors want to develop solid analytical skills and good programming habits in their students. While course length and hours may not permit all the topics in the book to be covered, the teacher should encourage students to read and study any topics not covered in class.

Secondly, it is assumed that teachers using this book want to prepare students to use the tools available to develop structured programs. Although flowcharts are used throughout the text, students are introduced to other tools such as pseudocode, HIPO charts, and Nassi-Shneiderman charts. Each teacher can determine what method should be stressed.

Thirdly, it is assumed that teachers using this book have heavy demands placed upon the use of their time. A great deal of time and effort are required to work with students, develop new courses, upgrade skills, and do the many tasks required to successfully teach courses in a high technology field. The material in this guide is intended to make teaching a comprehensive course in programming logic easier and less time-consuming.

MATERIALS PROVIDED IN THE INSTRUCTOR'S GUIDE

Every instructor has their own method of presenting material, using a text, and testing for mastery of material. Therefore, this guide attempts to accommodate a wide array of approaches to teaching. Suggestions are presented to facilitate efforts for tailoring this book to each individual's style. The following items are provided for each chapter of the text.

1. Comments and a brief outline. A brief overview and outline is included for each chapter. Often comments are included regarding the projects. The comments either explain the purpose of the project or give the rationale for the approach used in developing the solution.

2. Transparency masters for lecture outlines. After transparencies for the overhead projector are made from the masters, examples and lecture notes can be jotted on the masters. The use of the topic transparencies assist you in organizing your presentation and provide a checklist for determining if the material covered in the chapter has been presented.

   The use of the topical transparencies also assist your students. New terms and concepts are visually, as well as verbally, presented. Since students will both hear and see the material, the concepts presented are mastered faster. The correct spelling of new terms is also visually presented as the material is discussed.

3. Answers to study guide questions. If you wish to have your students check their answers, overheads can be made from the solutions. For your convenience, both the questions and the answers are provided.

4. Chapter tests. A test is provided for each chapter. Since most of the questions are objective, students can record most of their answers on a scan sheet and the tests can be scored electronically.

5. Solution for projects. Depending upon the characteristics of the project, different formats are followed in presenting the solution. Sometimes additional objectives are given for a project or the rationale used in developing a solution is explained. In other cases, suggested activities are given for the project.

   If ten analysts were assigned to do each of the problems, ten solutions would be developed for each problem. A programming problem usually has more than one solution. You may wish to modify some of the solutions provided in this guide. Or when using the solutions in class, you may wish to suggest some of the other alternatives available.

   VTOCs and flowcharts are provided for all of the problems. If you want your students to develop other types of logic plans, the solution given in the text can easily be converted to pseudocode, HIPO charts, or to Nassi-Shneiderman charts. However, the solution for project 4-1 does provide solutions using pseudocode, HIPO charts, and Nassi-Shneiderman charts.

TEACHING SUGGESTIONS

When PROGRAMMING LOGIC was field tested, students had little difficulty in developing detailed logic plans for the projects. Since the projects for the first two chapters are designed to reinforce the concepts presented in the text and to guide the students' thinking, students experienced success in completing their first assignments. Later when complete problems were assigned, students understood what was required and had confidence (because they were successful in their previous assignments). Students once again achieved a high degree of success in completing assignments.

A student reviewer commented "students will learn more and do a better job on the case study if the instructors make them read the entire chapter and do the study guides before they start the problems. It will give them (the students) a better understanding of the problems and the logic involved in each problem." Our experience in field testing the material supported the student's statement.

When field testing the material, each problem was assigned in phases. In the first phase students answered the questions related to the problem, listed the tasks that must be performed, and developed a VTOC. In class, various approaches to solving the problem were discussed. Before the students were assigned the task of developing a detailed logic plan for a problem, the VTOC to be used was agreed upon. After the logic plans were completed, overheads were used that provided a suggested solution so that students had immediate feedback regarding the correctness of their solutions.

HOW MIGHT THE MATERIAL BE PRESENTED

The time allocated to each of the eight chapters will differ depending upon your individual course objectives and the background of your students.

SUGGESTED PLAN FOR A 16-WEEK TWO SEMESTER HOUR COURSE

Chapters	Projects	Time	Material Covered
1-2	All	3 weeks	Although your students may have been exposed to some of the material presented in the first two chapters, reviewing the material helps to give your students a common background regardless of what prior experience they have had in data processing. The projects are relatively short and can be completed in a reasonable amount of time.
3	All	2 weeks	Project 1 requires students to answer questions designed to guide their thinking, to develop a VTOC, and to develop a detailed logic plan. The mini projects are designed to emphasize the correct use of indicators and to show the relationship that exists between a detailed logic plan and source code.
4	All	2-3 weeks	Although identified as three separate projects, when combined the three projects require students to answer questions over the program specifications, to develop a list of tasks to be performed, and to develop a detailed logic plan. Each instructor decides what technique should be used in developing the detailed logic plan.
5	1,2,3,4 5 optional	2 weeks	Projects 1 through 4 are all related to the savings update program described in Chapter 5. Project 5 is relatively short and provides a good review of many of the concepts presented in the previous 4 chapters.
6	1	2 weeks	The timecard edit program is directly related to the concepts presented in Chapter 6. Students must study the background information presented for the payroll system in order to determine how to edit the time-card. Project 6-2 would make an excellent take-home final exam.
7	1,2	2 weeks	Both problems are relatively short and reinforce the concepts presented in Chapter 7.
8	Select 2 programs	2-3 weeks	Some time should be spent discussing the total accounts receivable system. Although specifications are provided for five programs, only two programs should be assigned.

The suggested chapter tests can be combined into unit tests. If you prefer to make up your own tests, the ones provided in this guide can be used in alternating semesters as study guides. One semester the study guides provided in the text could be used. The following semester the objective tests could be used as study guides.

SUGGESTED PLAN FOR A 16-WEEK THREE SEMESTER HOUR COURSE

Since more time will be spent in class each week, the material can be covered faster than when the material is used for a two credit hour course.

Chapters	Projects	Time	Material Covered
1-2	All	2 weeks	Upon completing the two chapters, students will have the common core of knowledge required to develop structured programs.
3	All	1 week	Concepts presented in Chapters 1 and 2 are reinforced. The honors list program is illustrated from its inception to its implementation.
4	All	2 weeks	Using an example that requires the use of online terminals, students are introduced to the advantages and disadvantages of using flowcharts, HIPO charts, Nassi-Shneiderman charts, pseudocode, and decision tables.
8		1 week	The accounts receivable case study can be introduced. Some of the problems can be assigned as the last three chapters are covered or all of the five programs can be assigned during the last four weeks.
5	All	2 weeks	In Chapter 5 file handling, indicators, IF/THEN/ELSE, and PERFORMS are reviewed. Payroll examples illustrate the sequential processing of multiple files, random access of records, and the updating of a master file.
6	All	2 weeks	Some of the basic techniques used in editing data are covered.
7	All	2 weeks	Basic concepts regarding the use of print files are reviewed and then covered in more depth.
8	1-5	4 weeks	Many of the techniques presented in the first seven chapters must be used to develop the logic for the accounts receivable programs.

During the last four weeks the specifications for other programs in the accounts receivable system can also be developed. Students can work in groups and use design walkthroughs to determine the validity of their detailed logic plans.

SUGGESTED PLAN FOR A 16-WEEK ONE SEMESTER HOUR COURSE

Although there is more material that can be covered in a one semester hour course, it is still recommended that all seven chapters be covered. No more than two or three projects that require the logic for a complete program to be developed should be assigned. The following plan might be considered.

Chapters	Projects	Time
1	All	2 weeks
2	All	2 weeks
3	All	2 weeks
4	All	3 weeks
5	5	2 weeks
6	1	2 weeks
7	Study guide only	2 weeks
	Final exam	1 week

ADDITIONAL USE OF THE MATERIAL

Chapter 8, the accounts receivable case study, can be used as the problem set for an advanced programming course or for an independent study. When used for an independent study, specifications can be developed for more of the programs detailed in the accounts receivable system overview. Two or three students can work together on the project and incorporate many of the concepts covered throughout the text into the design, implementation, and documentation of the programs.

DEVELOPING A NEW COURSE?

If you would like a copy of the course justification which includes the course description, rationale for including the course in the curriculum, and the course objectives, feel free to contact:

Marjorie Leeson
Delta College
University Center, MI 48710
(517) 686-9134

YOUR SUGGESTIONS ARE INVITED

Many different styles of teaching exist among the faculty members who will use this material. Regardless of the delivery system used by the instructor, SRA and the author feel the material provided in this guide will help both the veteran and novice programming logic instructor. In return, we hope you will share suggestions for improving the text or this guide. Please submit projects or techniques that might be incorporated in future editions. Any suggestions you might have should be submitted to:

Data Processing Editor
SRA, Inc., College Division
1540 Page Mill Road
P.O. Box 10021
Palo Alto, CA 94303

CONTENTS

CHAPTER

INTRODUCTION TO PROGRAM DESIGN AND DEVELOPMENT CHAPTER 1

Many terms used in electronic data processing are introduced. While some students may be familiar with some of the terms, other students may not know the meaning of many of the technical terms. The importance of learning the meaning of terms associated with data processing and developing logic should be stressed. The new terms introduced in each chapter are listed at the end of the chapter under the caption Key Words. These terms are defined in the glossary.

After reading Chapter 1 and completing the required assignments, your students should be able to accomplish the tasks listed under the title Student Objectives. Throughout each chapter are Checkpoint questions. Students should be encouraged to answer the questions and check their answers with the ones found in the back of the text.

Although each instructor will present the material covered in each chapter differently, the following material should be stressed.

I. The problems created within the installation when designing and programming standards are not developed and followed.

II. The reasons why computerized projects were often overbudget and behind schedule.

III. The reasons why management is concerned with the cost of maintaining programs.

IV. The criteria used for evaluating a structured program.

V. The reasons why a program should be structured and modular.

VI. The steps in designing and completing a program.

   A. Definition of the problem. The program specifications must be completely defined. The Design Checklist can be used to determine if the specifications are complete.

   B. The I/O needs and formats must be determined.

   C. The internal and external controls needed to determine the validity of the processing and output must be established.

   D. A detailed logic plan must be developed.

      1. The tasks required to process the input must be determined.

      2. The tasks should be organized into groups or functional modules.

      3. A VTOC should be constructed that shows the relationships that exist between the various modules.

      4. For each module a detailed logic plan should be developed.

   E. A design walkthrough must be conducted to test the logic plan. The logic plan should be in nontechnical terms and language independent.

   F. The logic plan is converted to source code.

   G. After a clean copy of the source code is available, a code walkthrough is conducted.

H. The program is compiled and tested. The plan must include testing with

1. clean data

2. data that tests each error routine

3. no data

I. Before the program is considered operational, the internal and external documentation should be completed.

J. As soon as the program is operational, an evaluation must be made to determine if the design objectives were met.

VIII. The building-block concept. The VTOC used for the loan program should be presented as a foundation used to develop all future programs. The basic functions of the seven modules should be stressed.

Chapter I introduces students to several tools of the structured programmer. Since these tools and basic concepts are used whenever a structured program is developed, students are introduced to the concept in this chapter. The concepts are later presented in more detail, reinforced in a variety of ways, and then used in additional illustrations and projects. The concepts that fall into this category are the use of

indicators	editing
DO WHILE	files
PERFORM. . .UNTIL	VTOCs
sequence	IF/THEN/ELSE
iteration	design checklists
selection	walkthroughs
internal controls	standards

The following projects can be assigned and are designed to provide positive reinforcement of the following concepts:

Project 1	Illustrates use of the seven building blocks presented in the chapter. Students are to group the tasks identified under the correct module.
Project 2	Test data is used in conducting a design walkthrough. Students should be able to identify the error in the program design and the omission of a field from the test data.
Project 3	Emphasizes the similarities that exist between two report-producing programs. The students are also asked to do a flow-chart for the process records module.

CHAPTER 1 PAGE 1

I. WEINBERG
    A. RESIST THE URGE TO CODE
    B. EGOLESS PROGRAMMING

II. PROGRAMS SHOULD BE
    A. DESIGNED
    B. THE DESIGN TESTED FOR:
        1. LOGIC ERRORS
        2. OMISSIONS
    C. CODED
    D. DEBUGGED

III. DEVELOPMENT OF PROGRAMMING CONCEPTS
    A. PROGRAMMING IN ITS INFANCY
        1. STANDARDS NOT AVAILABLE
        2. ONLY VENDORS' SCHOOLS
        3. CRITERIA FOR SUCCESS
            A. JOB DIDN'T ABORT
            B. VALID OUTPUT
        4. JOBS NOT DESIGNED -- FIRE-FIGHTING
            A. MANAGEMENT WANTED IMMEDIATE RESULTS
            B. COMPUTERS MUST NOT BE IDLE
    B. EARLY DESIGN CONCEPTS
        1. PROGRAMS VIEWED AS AN ARTISTIC CREATION
        2. PROGRAMS WERE OFTEN OVERLY COMPLEX
    C. POORLY DESIGNED PROGRAMS RESULTED IN:
        1. LACK OF PRODUCTIVITY
            A. 1965 STUDY 10-12 LINES OF DEBUGGED CODE PER DAY
            B. 1975 STUDY PRODUCED SAME RESULTS
        2. HIGH COST OF MAINTAINING PROGRAMS

- D. IMPACT OF VIRTUAL STORAGE
  - 1. THRASHING
  - 2. LACK OF THROUGHPUT

IV. SOLUTIONS BEGIN TO EMERGE - STRUCTURED DESIGN

- A. BASIC CONTROL STRUCTURES - BOHM - JACOPINI
  - 1. SEQUENCE
  - 2. ITERATION - LOOPING
    - A. DO WHILE
    - B. PERFORM. . .UNTIL
  - 3. SELECTION - IF/THEN/ELSE
- B. ELIMINATION OF GO TOs - RATIONALE

V. CRITERIA FOR A STRUCTURED PROGRAM

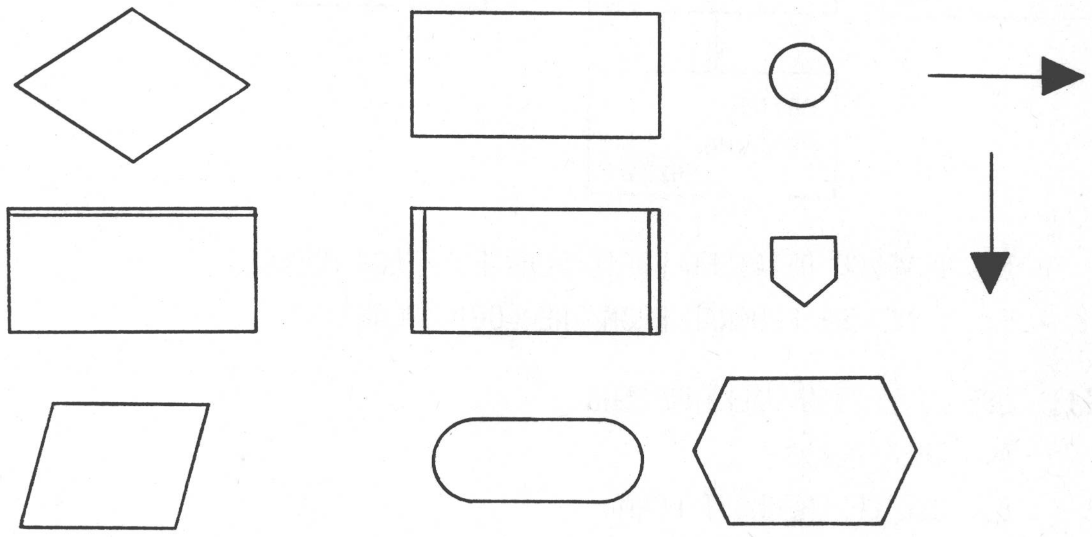
- A. MODULAR - ONE TASK OR FUNCTION PER MODULE
- B. WELL-DESIGNED - TOP DOWN
- C. EASY TO FOLLOW
- D. WELL FORMATTED - CONVENTIONS USED
- E. MEANINGFUL NAMES USED FOR FILES AND FIELDS
- F. SIMPLISTIC
- G. BASIC CONTROL STRUCTURES USED
- H. GO TOs AVOIDED
- I. INTERNALLY DOCUMENTED
- J. COMPLIANCE WITH INSTALLATION'S DESIGN AND PROGRAMMING STANDARDS

VI. ACCEPTANCE OF STRUCTURE CONCEPTS

- A. INCREASED PRODUCTIVITY -- 35 TO 65 LINES PER DAY OF DEBUGGED CODE

B. TEST CASE
   1. NEW YORK TIMES PROJECT
   2. 21 ERRORS IN 83,324 LINES OF CODE

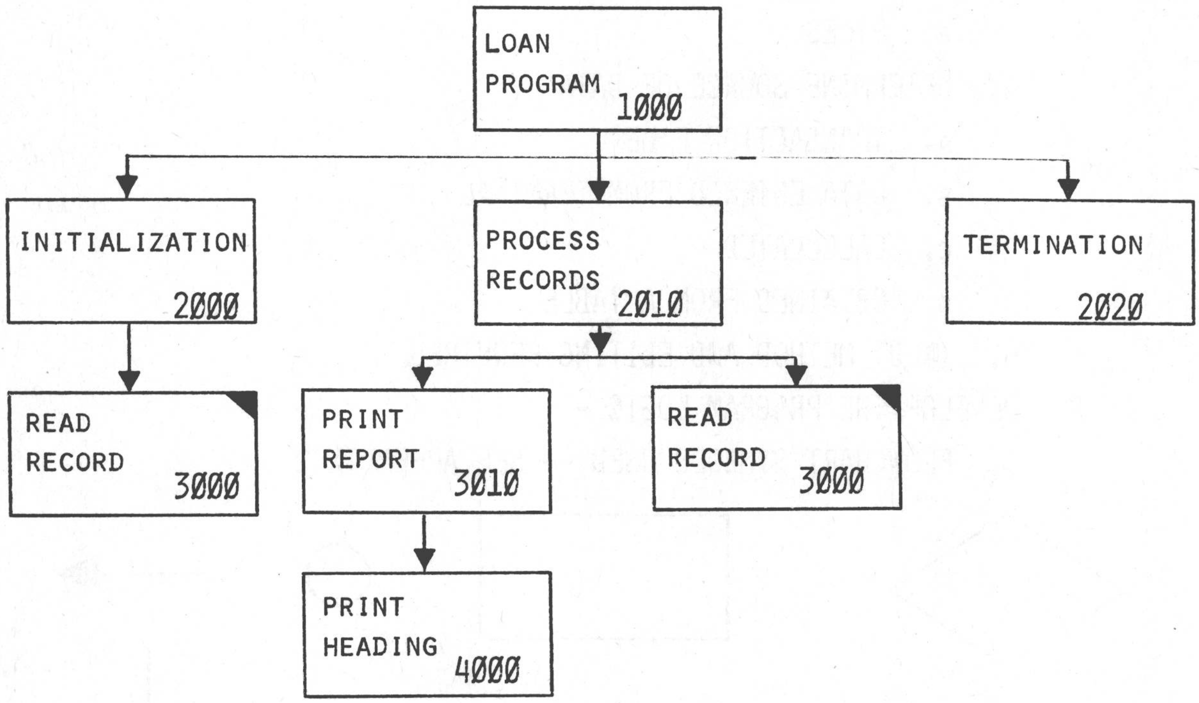
VII. STEPS IN DESIGNING AND COMPLETING A PROGRAM
   A. DEFINING THE PROBLEM
      1. DEVELOP SPECIFICATIONS - BATCH VS. ONLINE
      2. DETERMINE CONTROLS NEEDED
      3. DETERMINE I/O NEEDS AND FORMATS
         A. REPORTS
         B. FILES
      4. DETERMINE SOURCE OF DATA
         A. TRANSACTION FILE
         B. DATA ENTERED FROM TERMINAL
         C. CALCULATED
         D. OBTAINED FROM A TABLE
      5. INPUT METHOD AND EDITING REQUIRED
   B. DEVELOP THE PROGRAM LOGIC -
      1. FLOWCHART SYMBOLS USED -- SEE APPENDIX 2



2. MUST DETERMINE -- SEE DESIGN CHECKLIST
   - A. INPUT AVAILABLE
   - B. EDITING REQUIRED
   - C. CALCULATIONS REQUIRED
   - D. EXCEPTIONS OR DEVIATIONS
   - E. CONDITIONS UNDER WHICH DATA IS TO BE OUTPUTTED
   - F. CONTROLS REQUIRED

C. DEVELOPMENT OF A VTOC

EACH MODULE CAN BE CONSIDERED A BUILDING BLOCK!



D. DEVELOP DETAILED LOGIC PLAN FOR EACH MODULE

E. CODE THE PROGRAM FROM THE LOGIC PLAN

VIII. USE OF DESIGN WALKTHROUGHS
   - A. TEST PLANS
   - B. USE OF CHECKOUT FORM
   - C. INTENT OF WALKTHROUGH

IX. CODE WALKTHROUGH -- DETERMINE IF

A. CODE FOLLOWS LOGIC PLAN

B. ALL REQUIRED CODE IS PRESENT

C. ANY UNNECESSARY CODE HAS BEEN USED

D. INDICATORS, TOTALS, AND CONSTANTS ARE SET TO RIGHT VALUE

E. FIELDS WITHIN RECORDS ARE CORRECTLY DEFINED

F. HEADINGS, DETAIL, AND TOTAL LINES CORRESPOND TO THE PRINT LAYOUT FORM

G. ANY FIELD DEFAULTED

X. TESTING -- WITH

A. GOOD DATA

B. BAD DATA -- EACH ERROR ROUTINE MUST BE TESTED

C. NO DATA

XI. DEBUGGING PROGRAMS

A. NO REASON TO HAVE BUGS IF DESIGN AND CODE WALKTHROUGHS ARE COMPLETED

B. USE OF DEBUGGING AIDS

XII. DOCUMENTATION

A. INTERNAL

B. EXTERNAL

XIII. EVALUATION

XIV. RELATIONSHIP OF DESIGN TO CODE

A. DESIGN SHOULD BE AS LANGUAGE INDEPENDENT AS POSSIBLE

B. AMOUNT OF DETAIL REQUIRED DEPENDS UPON LANGUAGE BEING USED