

**Chapter
1**

**CODING
STRUCTURED COBOL
PROGRAMS**

*Get on the
Fast Track!*



TM

**SYS-ED/
Computer
Education
Techniques, Inc.**

Objectives

You will learn:

- Role and value of structured programming.
- The issues, problems, and shortcomings associated with spaghetti code.
- Three logic structures utilized in structured programming.
- Structured design.
- Structured walkthroughs.
- Structured flowcharts.
- The three nodes used in structured flowcharts.
- Pseudocode.

**1 Structured Programming
versus Spaghetti Code**

Structured programming is a technique for developing source code logic that is understandable and verifiably correct.

It employs a top-down refinement strategy in order to produce code built from a small set of logical constructs:

sequence	decision	loop
----------	----------	------

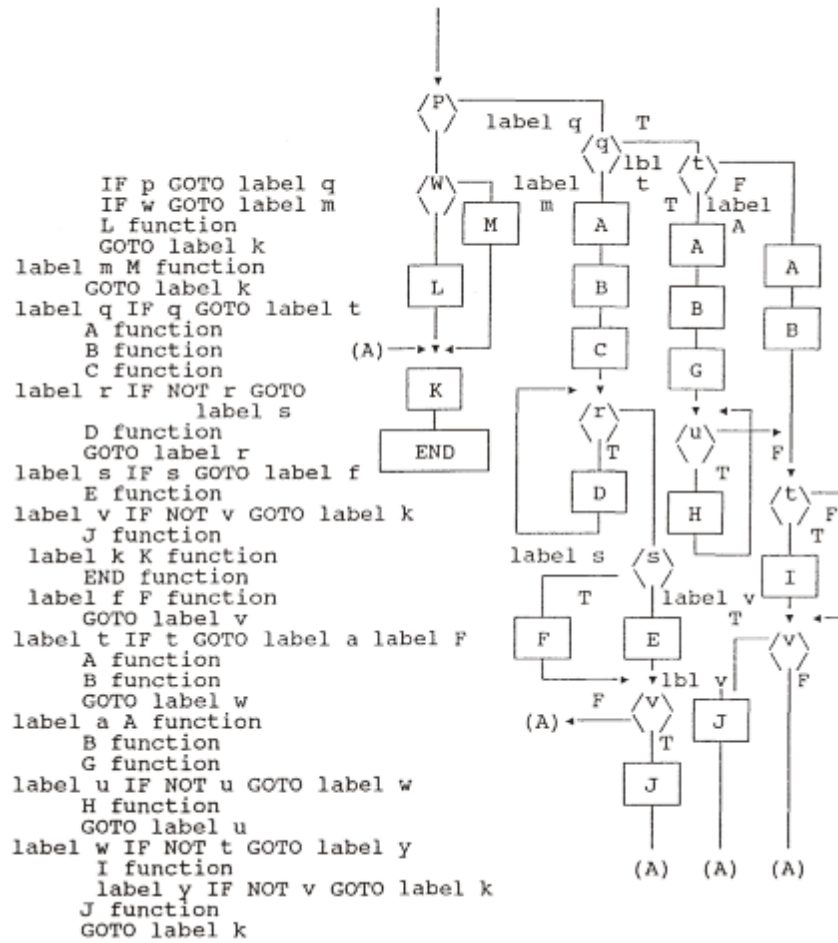
With structured programming, the design, coding, and testing of a program is comprised of interdependent parts within a well defined pattern of organization.

Traditional spaghetti-bowl programming is characterized by:

- Excessive use of unconditional branches and statement labels.
- A high number of GOTO statements which are being improperly utilized.

1.1 Example:
Spaghetti Code in a Unstructured Flowchart

(from IBM Independent Study Program, Structured Programming)



2 Three Eras of COBOL Program Development

There have been three eras of COBOL program development.

GOTO programming (circa 1965)

- Program designed using detailed program flowcharts.
- Extensive use of GOTO statements.
- COBOL code difficult to read and modify.

Modular programming (circa 1970)

- Sought to divide a program into a number of independent modules: one mainline and one or more subroutines.
- Increased use of PERFORM statements, but continued use of GOTO statements.

Structured programming (circa 1975)

- Program designed using a structure chart or other design document instead of a program flowchart.
- Use of three basic logic structures.
- Minimum use or complete absence of GOTO statements.

3 Structured Programming: Three Logic Structures

Structured programming has three fundamental logic structures:

Sequence	Selection	Iteration
	IF...ELSE	PERFORM...UNTIL

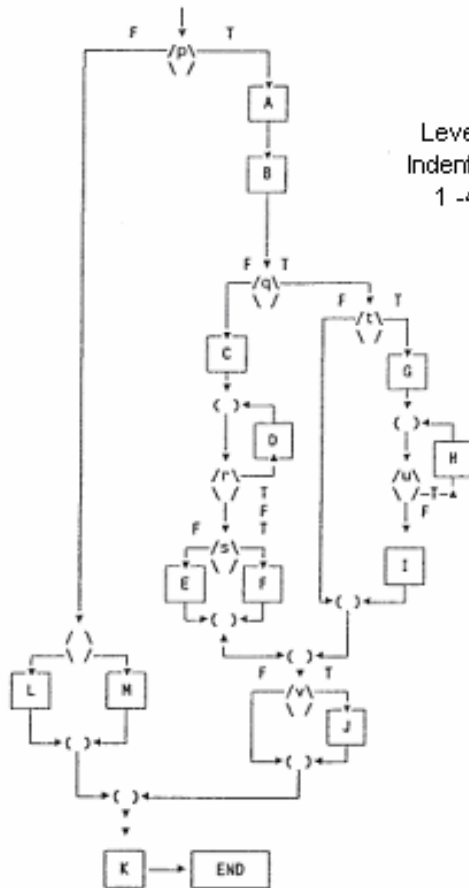
These three logic structures can be combined to produce the required logic.

The development of structured code built with these three logic structures results in:

- The entire program being a set of nested modules; each of which has one entry and one exit.
- Discouraging the use of the GOTO statement.
- The utilization of certain programming restrictions and conventions which promotes the implementation of proper structured programming techniques.
- Proper formatting and indentation.
- A program being subdivided into a small of modules; one-page modules are the general recommendation.
- No module being allowed to modify any other module.
- The use of top-down program design.

4 Example:
Structured Flowchart and Structured Code

(from IBM Independent Study Program, Structured Programming)



Level of Indentation:
1 -4

```

IF p THEN
  A function
  B FUNCTION
  IF q THEN
    IF t THEN
      G function
      DOWHILE U
      H function
      ENDDO
    I function
  (ELSE)
  ENDF
  ELSE
  C function
  DOWHILE P
  D function
  ENDDO
  IF s THEN
    F function
  ELSE
    E function
  ENDF
  IF v THEN
    J function
  (ELSE)
  ENDF
  ELSE
  IF w THEN
    M Function
  ELSE
    L FUNCTION
  ENDF
  ENDF
  K function
  END function
  
```

5 Improved Programming Methodologies

Structured programming is one of several methodologies used to produce better code which is easier to maintain.

The following techniques also have proven to increase data processing efficiency.

- Team operations.
- Top down program development.
- Code reading and walkthroughs.
- HIPO: Hierarchy Plus Input, Process, Output documentation technique.
- Program development support libraries.

6 Structured Programming: Benefits

The benefits of structured programming include:

- Producing programs that are easier to understand and maintain.
- Making it easier to test and debug code.
- Improving programmer productivity.
- Enhancing the readability of a program and making it easier to document.

7 Structured Programming: History

In 1966, Corrado Bohm and Guisepppe Jacopini proved that any program logic, no matter how complex, could be expressed using relatively simple structured program control elements.

Edsgar Dijkstra, a leader in structured programming, devised a set of ideas and series of examples for that clear thinking in the construction of programs.

Harlan D. Mills, mathematically proved that the theory and standards in programming procedures were practical and logical to implement.

Mills also highlighted the positive psychological impact of structured programming. Programmers were encouraged by the ability to write programs correctly and minimize careless errors.

8 Structured Design

Structured design is a set of program design techniques and guidelines used to specify the:

- functions in a program solution.
- data processed by each function.
- relationships among the functions.

Structured design is the development of a blueprint for a computer system which has the same components and interrelationships among the components as the original problem.

Structured design is used to define program solutions using single, well-defined functions that can be easily programmed, tested and maintained.

Structured design is not modular programming or top-down design; it is an extension of the two techniques.

9 Structured Walkthrough

A structured walkthrough is a series of reviews of a programmer's work by a peer group, each with different objectives, which occurs at a different time in the application development cycle. Aside from the programmer, common participants in the structured walkthrough include: reviewer, moderator, and secretary. The objective of a walkthrough is not to evaluate the programmers.

A structured walkthrough is used in conjunction with top-down design, coding, and testing.

The major benefits associated with structured walkthroughs include:

1. Early error detection
 - Detection is made during the design phase.
 - Errors are uncovered after the programmer has received the analyst's specifications and developed the logic to implement a solution.
 - Errors can be corrected during the coding stage.
2. Tracks progress in the application development cycle.

The criteria for measuring a successful walk-through include:

- Finding programming problems.
- Error detection.
- Reviewing everyone's work.

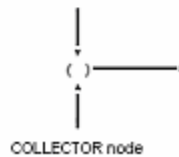
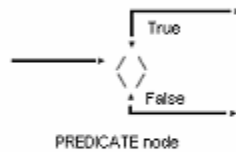
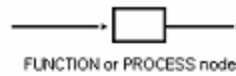
10 Structured Flowcharts

Structured flowcharts are:

- A visual representation of structured program logic.
- A tool for the programmer; however, it is not a substitute for proper documentation. This is because structured flowcharts are seldom updated.

Structured flowcharts have three elements or nodes:

Node	Explanation
FUNCTION or PROCESS	Represents a function or data transformation.
PREDICATE	Represents a test on data, resulting in either a true or false condition.
COLLECTOR	Represents a junction which always has two entries and one exit. It is needed only with the PERFORM...UNTIL statement and to rejoin the two paths of a decision box.

11 Flowchart: Three Nodes

These three basic nodes are combined into the three structured programming logic structures: sequence, selection, and iteration.

Any proper program can be written using three basic nodes.

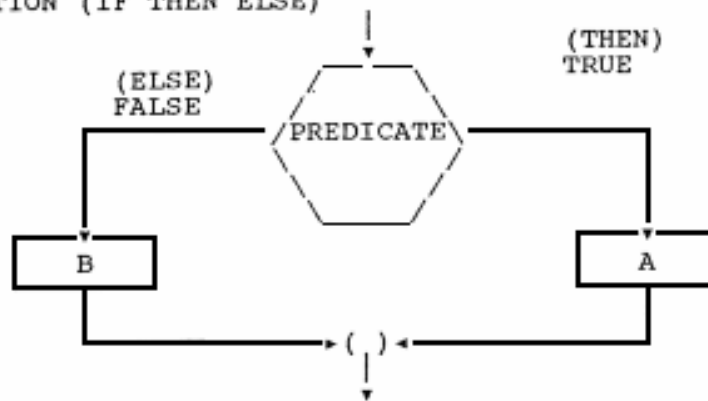
- A proper program has only a single entry point and one exit.
- For every node, there exists a path from the input line through that node to the output line; there is no infinite loop or dead code.
- The nodes can be replaced by pseudocode.

12 Structured Programming:
Three Basic Logic Structures

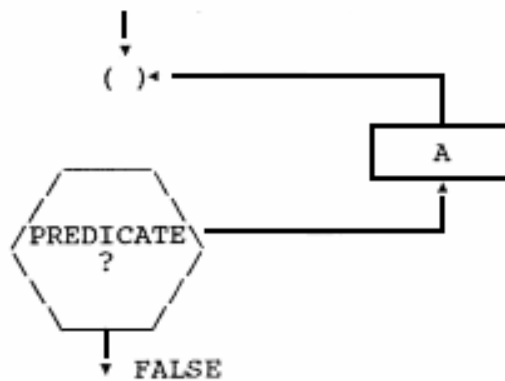
(a) SEQUENCE



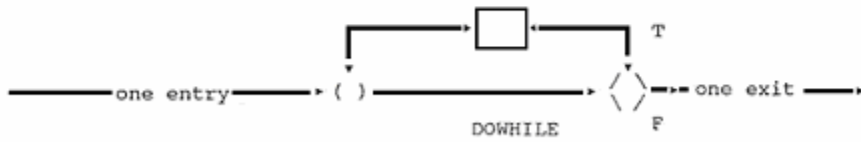
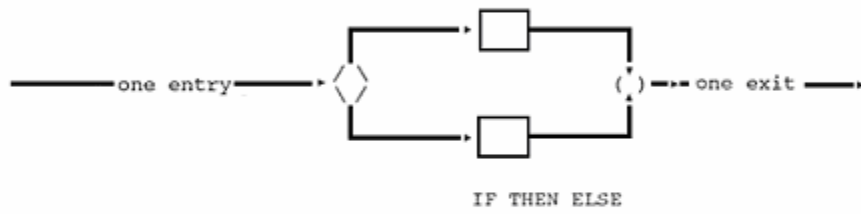
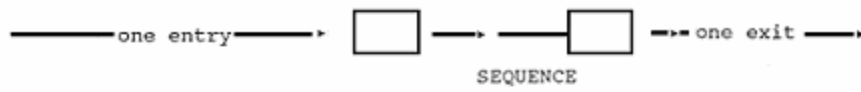
(b) SELECTION (IF THEN ELSE)



(c) ITERATION (DO WHILE)



13 Nodes Used in Well Written Programs



14 Pseudocode

The role and function of pseudocode is to:

- Serve as a design aid and documentation technique used in conjunction with structured programming.
- Use instructions similar to a computer language for describing program logic.
 - There are no precise syntactical rules or rules for indentation.
 - Indentation is used to represent levels of hierarchy for nested functions; it makes the logical structure more clear.
- Pair and utilize dowhile with enddo, if with else, and endif.
- Provide the functional equivalent of a flowchart.
- Allow the programmer to concentrate on logic instead of form.
- Provide an easier more flexible way to make update than with a flowchart.

14.1 Example: Pseudocode

```
Initialize
Open Files
Initial reads for OLD-MASTER and TRANSACTION files
PERFORM until no more data on both files
  IF OLD-MASTER social security number < TRANSACTION social security number
    PERFORM
      Copy old master record to new master record
      Set switch to read OLD-MASTER only
    ENDPERFORM
  ELSE IF OLD-MASTER social security number = TRANSACTION social security number
    PERFORM
      IF addition-write error indicating duplicate addition
      ELSE IF deletion-delete old master record
      ELSE IF salary-change-do salary update
      ENDIF
      Set switches to read both OLD-MASTER and TRANSACTION files
    ENDPERFORM
  ELSE IF OLD-MASTER social security number > TRANSACTION social security number
    PERFORM
      IF addition-add record to NEW-MASTER-FILE
      ELSE write error indicating no match
      ENDIF
      Set switch to read TRANSACTION-FILE only
    ENDPERFORM
  ENDIF
Read OLD-MASTER and/of TRANSACTION files as appropriate
ENDPERFORM
Close files
Stop run
```