



WICHITA STATE  
UNIVERSITY

UNIVERSITY LIBRARIES

## Program documentation at Wichita State University

Item Type	Article
Authors	Foley, David M.
Citation	Foley, D. M. (1983). Program documentation at Wichita State University. ACM SIGCSE Bulletin, 15(1), 133-136. <a href="https://doi.org/10.1145/952978.801034">https://doi.org/10.1145/952978.801034</a>
DOI	<a href="https://doi.org/10.1145/952978.801034">10.1145/952978.801034</a>
Publisher	ACM
Download date	2026-05-19 13:14:08
Link to Item	<a href="https://hdl.handle.net/10057/27501">https://hdl.handle.net/10057/27501</a>



PROGRAM DOCUMENTATION  
AT  
WICHITA STATE UNIVERSITY  
David M. Foley  
Wichita State University

INTRODUCTION: It has been widely recognized that formal documentation of computer systems has been a major problem for computer users because it is sometimes inaccurate, oftentimes incomplete, and occasionally even missing. This failure to adequately document software has led to higher production and maintenance costs, customer dissatisfaction and the development of useless programs. In a move to avoid these problems, the Computer Science Department at WSU has established a standard for program submittal in all its classes with laboratories. This standard also promotes good programming practices, such as designing before coding and modularization, provides our students with a disciplined approach, orients them to what industry requires and facilitates the grading of their programming assignments. The standard is introduced in CS200, Introduction to Programming, in which the focus is problem analysis and the writing of the solution in pseudocode.

This standard was implemented in August, 1979, along with a new computer science program that was based on the Curriculum 78 guidelines. Since that time the standard, which was originally developed by then-faculty Don Newman and Steve Taylor, has undergone a few revisions and was updated most recently by Peggy Wright, a computer science graduate student.

APPLICATION OF THE STANDARD: The documentation standard described here shall apply to all computer science courses containing laboratory sessions where programming work is assigned. These courses include all the programming languages and the upper-division classes such as Fundamental Algorithms, Computer Organization and Programming, and File Processing Techniques.

ORGANIZATION OF THE STANDARD DESCRIPTION: This standard encompasses both physical and logical aspects of the documentation. Certain materials

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

© 1983 ACM 0-89791-091-5/83/002/0133 \$00.75

described below are to be assembled in a prescribed manner to form the documentation package which will be submitted and graded in fulfillment of the requirements for the laboratory. These materials also constitute a logical structure which guides the student's efforts from the problem definition through solution and implementation to program verification and culmination of the programming effort.

CONTENT OF THE DOCUMENTATION: The materials comprising the documentation must cover the four phases of problem solution -- analysis, programming, coding, and verification. In the analysis phase the problem is defined and the high level documentation is produced. In the programming phase, a solution to the problem is planned and detailed to produce the intermediate level documentation. During the coding phase the pseudocode program, description of data, and the modular structure are converted to program code. Program comments, control flow charts, and decision tables are used to further explain the code. In the verification phase, the program is operated to test whether it accurately produced the results specified in the problem definition. Difficulties found in the test run are corrected through debugging and providing directions to the operator. This process is recorded in run documentation.

Outline of Contents for Standard Documentation Folders

- A. High Level Documentation
  - 1. Description of the Problem
  - 2. Input Information
  - 3. Processing Information
  - 4. Output Information
  - 5. Sample Operations
- B. Intermediate Level Documentation
  - 1. Structure Chart
  - 2. Internal Data Structures
  - 3. Pseudocode
  - 4. Module Interface Tables
- C. Low Level Documentation
  - 1. Program Code
  - 2. Comments
- D. Run Level Documentation
  - 1. Test Data
  - 2. Test Results

\* \* \*

## A. HIGH LEVEL DOCUMENTATION

The purpose of this section of the documentation is to define the problem and what would be a solution, with sufficient detail so that the solution can be planned.

### 1. Description of the Problem

Name - give a short title.

User Application - tell what needs to be done.

Purpose of Program - give the objectives in terms of how this input, processing, and output sequence will satisfy the needs addressed above.

### 2. Input Information

Repeat the following for each input:

Name: (Give the name of the input file or parameter.)

Type - tell what medium the data is on and any important physical details.

Format - give the format(s) of data structure(s) (usually record layout or table) including data element names and positions.

Data Elements - tell what each element of data means, how it is restricted, and what data type it is.

Organization - (optional) explain how the structures are sequenced and/or arranged in the file or area.

### 3. Processing Information

Repeat the following for each calculation:

Name: (Give a calculation name.)

Conditions - (optional) tell under what conditions the calculation is performed.

Steps - give the steps of the calculation.

Special Detail - (optional) give algorithms, formulas, or constants to be used.

### 4. Output Information

Repeat the following for each output:

Name: (Give the name of the output file or parameter.)

Type - tell what medium the data is on and any important physical details.

Format - give the format of the data records (usually record layout or print chart).

Data Elements - tell what each element of data means and how it is formulated.

Organization - (optional) explain any unusual feature of the sequence arrangement of the output.

### 5. Sample Operations

Provide hand-printed or typed materials intended to demonstrate the operation of the program. This material is created in advance of the design of the program and is not sufficient to test the program.

Input - give specific input data including possibly both correct and incorrect data.

Output - show the output which should result from the input.

## B. INTERMEDIATE LEVEL DOCUMENTATION

The purpose of this section of the documentation is to describe the plan for the solution of the problem. For a program containing more than one program module this includes a chart of the program structure (Structure Chart) and a description of the interaction of the various modules (Module Interface Table). These two items are not required for programs consisting of a single module. For any program descriptions of the data used (internal data structures) and problem solution (pseudocode) are required.

### 1. Structure Chart

Show the tree diagram of the modules in the program, indicating the interaction of the modules. Label the modules with the same names used in the pseudocode.

### 2. Internal Data Structures

For each program module this section should list and define the names, structure type and initial contents of all storage areas used to hold data between input and output. It should give the values, meanings and representations of any non standard data types. For a given module the list of names to be included here can be derived as follows: List all data names used in the module and delete from this list all input and output variables. The remaining list contains the internal data structures.

### 3. Pseudocode

This section should describe, in an easily readable form, how the module will solve the given problem. The term "pseudocode" is not intended to refer to a precise form of expression. Rather, it refers to using standard English terms in a restricted manner (but without lots of formal rules) to describe the algorithmic process involved. Good pseudocode should employ a restricted subset of English in such a way that it resembles a good high level programming language. The pseudocode description of the problem should state the problem solution so clearly that it can easily be translated into the programming language to be used.

### 4. Module Interface Tables

This includes a list of all modules, giving the name of each module and a brief (one sentence) description of its function or purpose. Also included is a description of each "interface" between modules. This module interface information can be shown in a table of "inputs" and "outputs", one table for each module interface. This information can be created as follows.

As you are writing the specifics of a module (the "parent" module), and you need to use another module (the "child" module) write down the module interface for this pair of modules. This module interface includes:

- i) the name of the modules
- ii) the child module's functional definition or purpose

- iii) a list of what the child module is to produce (its outputs) for the parent module and
- iv) a list of what the child module needs (its inputs) from the parent module in order to produce the outputs.

The module interface tables then show the inputs and outputs for each module interface.

C. LOW LEVEL DOCUMENTATION

The purpose of this documentation is to present an executable version of the program, along with the information needed to clarify how it has been encoded and what is required to test it.

1. Program Code (Source Program)  
The source program listing provided by the compiler is required here.
2. Comments  
Minimally adequate comments should be included in the source program listing to explain the relation between the program code and the preceding documentation. The comments should be used only when the encoding or translation is not the obvious one. When more than one line of program code represents a single item in the preceding documentation, these lines should be separated through use of indentation or with a blank line above and below.

D. RUN LEVEL DOCUMENTATION

The purpose of this documentation is to demonstrate the operation of the program.

1. Test Data  
Include a listing of input data for the program which will cause every statement to be executed at least once. (The sample inputs may be used as part of the test data, but they are not generally sufficient for the test.) Each line or group of lines should be identified according to what it tests.
2. Test Results  
Include an output listing, attached to the program listing (from a single job on the computer), showing the results of running the program with the Test Data.

\* \* \*

CS200            Sample Problem            10-10-82

HIGH LEVEL DOCUMENTATION

Description

Name - Inventory Costing

User Application - The manager of the warehouse needs assistance getting a report to the president of Teak Limited each week with an up-

to-date value of inventory. It is necessary to show the value of every item in the warehouse as well as the total. In a week's time only a few items change in the warehouse and only the changed data can be gathered readily.

Purpose - The report will provide the information required for the whole warehouse by correcting only the changed input cards and the date card. The output will include a list of every item with the raw data and its contribution to the value of the inventory. A total will appear only at the end of the report. The date will appear on each page to indicate when the data was current.

Input Information

Date card:  
Type - 80 column punched cards

CC	Field	Form
1-8	--	blank
9	card code	'I'
10-24	part-number	
25-26	--	blank
27-52	item-name	
53-61	--	blank
62-66	unit-price	3 decimal places
67-74	--	blank
75-80	quantity	

Data elements -  
part-number - id number of each item.  
item-name - description of each item.  
unit-price - price of one unit in dollars.  
quantity - number of units in the warehouse.

Organization - all cards follow the date card in the deck in the order they will appear in the listing.

Processing Information

Value of Inventory Calculation -  
Condition - used for all items whose numbers do not begin with an '8'. (Items beginning with '8' have no value.)  
Detail - value = unit-price x quantity rounded to a whole dollar.

Output Information

Inventory valuation -  
Type - printed paper  
Format - (use a report layout sheet for this)  
Data elements -

Number	Title/Heading	Source of data
1	PART NUMBER	Input part-number
2	ITEM NAME	Input item-name
3	UNIT PRICE	Input unit-price
4	QTY	Input quantity

5	AMOUNT	Value of inventory as described in the Processing Information
6	TOTAL INVENTORY VALUE	Sum of AMOUNTs
7	TOTAL INVENTORY ITEMS	Count of part nums.
8	PAGE	Page number
9		Input date

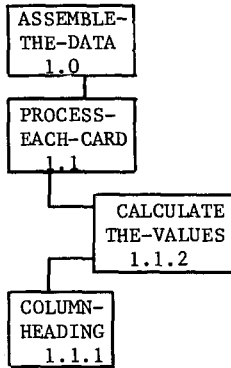
Organization - sequence is determined by the input

Sample Operation

(Sample input and output may be shown on a report layout sheet.)

INTERMEDIATE LEVEL DOCUMENTATION

Structure Chart



Note: In a structure chart each box represents a module of a program. The name of the module is inside. The box below that is connected with a vertical line is a submodule of the upper box.

Internal Data Structures

- Out-of-cards-flag - binary (3 chars)
  - Page-counter - numeric (2 digits)
  - Line-counter - numeric (2 digits)
  - Total-items - numeric (3 digits)
  - Total-value - numeric (12 digits)
  - Report-lines - alphanumeric (up to 133 chars)
- 9 different line formats are used for the report heading, body and totals.

Pseudocode

Assemble-the-data.  
 Open all files.  
 Read a card.  
 If there is no card, set flag to indicate out-of-cards.  
 Until out of cards, repeat process-each-card.  
 Repeat write under-line 2 times.  
 Assemble and write total value line.  
 Assemble and write total items line.  
 Close all files.  
 Stop.

Process-each-card.  
 If the card code is D,  
 move the date to the date line and do the column-heading.  
 If the card code is I,  
 calculate-the-values.  
 Read a card.  
 If there is no card,  
 set flag to indicate out-of-cards.

Column-heading.  
 Assemble and write the 3 top report heading lines on the top of a page, space down, assemble and write the two remaining lines and space down.  
 Set the line counter to 7.  
 Add 1 to the page number.

Calculate-the-values.  
 Clear out the report body line and assemble the data from the input line into it.  
 Calculate the amount as price times quantity.  
 Add the amount to the total value.  
 If the line counter indicates there is no room on the page, do a column-heading.  
 Write the report body line.  
 Add 1 to total items and line counter.

Module Interface Tables

List of modules

Number	Name	Function
1.0	Assemble-the-data	Read data cards; process and write report
1.1	Process-each-card	Process card or set flag if out of cards
1.1.2	Calculate-the-values	Calculate item values and accumulate totals
1.1.1	Column-heading	Write report headings

Interfaces between modules

	Inputs	Outputs
1.0	Out-of-cards-flag D I Date	Out-of-cards-flag Total-value Total-items Line-counter
1.1	Unit-price Quantity Total-value Total-items Line-counter	
1.1	Part-number Item-name Date Unit-price	Total-value Total-items Line-counter
1.1.2	Quantity Total-value Total-items Line-counter	
1.1.1	Date	Line-counter
1.1.2	Date	Line-counter

REFERENCES

1. Bohl, M. A Guide for Programmers, Prentice-Hall, 1978.
2. Tools for Structured Design, Science Research Associates, 1978.
3. Myers, G. J. Composite/Structured Design, Van Nostrand Reinhold, 1978.