AD-A034 439 ARMY TRAINING SUPPORT ACTIVITY FORT GORDON GA COMPUT--ETC F/G 9/2
PROJECT ABACUS HANDBOOK ON PROGRAM DOCUMENTATION AND DEBUGGING.(U)
FEB 76 W L DUNCAN

UNCLASSIFIED CTSD-TR-76-1 NL

1 OF 1
AD-A
034 439

END
DATE
FILMED
2-15-77
NTIS

PROJECT ABACUS HANDBOOK ON PROGRAM DOCUMENTATION AND DEBUGGING

ARMY TRAINING SUPPORT ACTIVITY
FORT GORDON, GEORGIA

16 FEBRUARY 1976

REPORT CTSD - TR - 76 - 1

AD_____

PROJECT ABACUS HANDBOOK ON PROGRAM DOCUMENTATION AND DEBUGGING

William L. Duncan

Technical Applications Division
Computerized Training Systems Directorate
US ARMY TRAINING SUPPORT ACTIVITY
Fort Gordon, Georgia 30905

16 February 1976

Approved for public release:
Distribution unlimited

Prepared for:
US ARMY TRAINING AND DOCTRINE COMMAND
Fort Monroe, Virginia 23651

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*
US Army Training Support Activity
Computerized Training Systems Directorate
Project ABACUS, Fort Gordon, GA 30905

2a. REPORT SECURITY CLASSIFICATION
UNCLASSIFIED

2b. GROUP

3. REPORT TITLE
Project ABACUS Handbook on Program Documentation and Debugging

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
Technical Report

5. AUTHOR(S) *(First name, middle initial, last name)*
William L. Duncan

6. REPORT DATE
16 February 1976

7a. TOTAL NO. OF PAGES

7b. NO. OF REFS

8a. CONTRACT OR GRANT NO.

b. PROJECT NO.

c.

d.

9a. ORIGINATOR'S REPORT NUMBER(S)

9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*

10. DISTRIBUTION STATEMENT
Distribution of the document is unlimited.

11. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY
US Army Training and Doctrine Command
Fort Monroe, Virginia 23651

13. ABSTRACT

Two of the most important elements of a computer programming effort are program debugging and documentation. This report provides an outline of fundamental debugging techniques to be used by Project ABACUS programmers. These are proven techniques that have been developed through experience.

Documentation guidelines, based on US Army Regulations, have been established in this report to provide the programmers with standards for program documentation. The development of standardized procedures and thorough program documentation will enable succeeding programmers to assume responsibility for program maintenance expeditiously and with a minimum of confusion.

DD FORM 1 NOV 68 1473 REPLACES DD FORM 1473, 1 JAN 64, WHICH IS OBSOLETE FOR ARMY USE.

UNCLASSIFIED

Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Computerized Training System						
Project ABACUS						
Computer Programming						
Computer Programming Documentation						
Documentation						
Debugging						
Computer Program Debugging						
Computer Program Folder						
Computer Programming Techniques						
Computer Debugging Techniques						

NOTICES

This report has been reviewed and is approved.

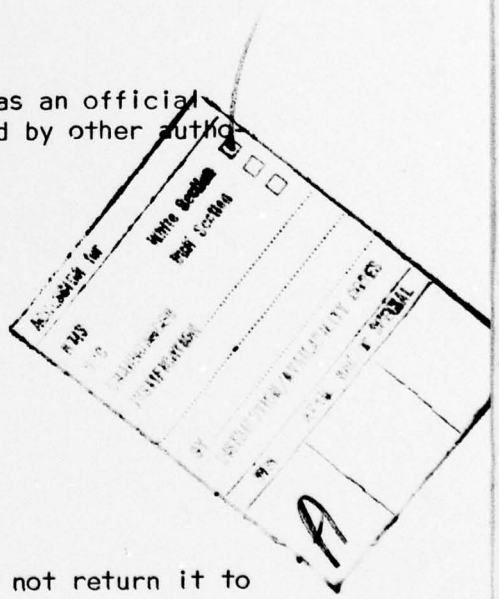
Frank E. Giunti

FRANK E. GIUNTI
Director, Computerized Training
Systems Directorate

G. B. HOWARD
Colonel, Signal Corps
Product Manager, Computerized
Training Systems Directorate

Disclaimer

The contents of this report are not to be construed as an official Department of the Army positions unless so designated by other authorized documents.



Disposition

Destroy this report when it is no longer needed. Do not return it to the originator.

FOREWORD

Two of the most important elements of a computer programming effort are documentation and program debugging. Documentation is often neglected and debugging done in a random manner. In order to avoid these pitfalls in the programming effort of Project ABACUS, a handbook was prepared for use as a guide in program documentation and debugging efforts. This handbook incorporates the pertinent Army Regulations and on-the-job experiences.

ROBERT G. FOSTER
LTC, SigC
Program Director, Project ABACUS

TABLE OF CONTENTS

I. INTRODUCTION

This manual sets forth procedures and techniques for documenting, modifying, and debugging the application and system programs being developed for Project ABACUS.

II. STANDARD OPERATING PROCEDURE (SOP) FOR DOCUMENTATION

A. Purpose. This section establishes standards and provides guidance for preparing documentation of computer programs. Documentation of computer programs is required to:

1. Facilitate program use and maintenance.

2. Reduce the impact of personnel changes.

3. Permit flexibility in reassignment of programmers to meet unusual workloads.

4. Contribute to the exchange of programs between installations to minimize duplication of effort.

5. Facilitate conversion to other Automatic Data Processing Equipment (ADPE).

6. Assist in audit of computer processing.

Documentation is often known as the "Achilles Heel" of any Data Processing (DP) department. For this reason, a continuous update of documentation is also needed to prevent lagging. Changes and dates of revision should be maintained, as a programmer or analyst is likely to forget the intricacies and vital workings of a program or a procedure.

B. Computer Program Folder. A computer program folder will be established for each computer program prepared for operation. Computer program folders will be prepared in accordance with standard outline and instructions indicated in Appendix J of Army Regulation, AR 18-7, Data Processing Installation Management, Procedures, and Standards.

Format of Outline:

Chapter 1 - General

1. Identification
2. Description
3. Environment

Chapter 2 - Specifications

1. Purpose

2. Forms

3. Other Information

Chapter 3 - Program Maintenance

1. Operating Environment

2. Assumptions

3. Input

4. Output

5. Subroutines

6. Addenda

Chapter 4 - Operating Instructions

1. General

2. Input

3. Output

4. Setup

5. Termination

6. Addenda

Exceptions to the requirement will be program folders covering portions of a real time system, and subsets of an overall program system may deviate from the standard outline, as appropriate, due to their uniqueness.

C. Source Program Documentation.

1. Document as much of the program within the code as possible by extensive use of meaningful comment notation imbedded within the coding. This will concentrate the documentation where it will do the most good. If the documentation remains with the program, comments and code maybe modified at the same time. All comment coding will start with a blank comment line and end with a blank comment line.

2. All statement numbers will be sequential.

3. The leading information that must be present at the beginning of the program is:

```
C
C Program Name:
C Written By:
C Date Written:
C Purpose:
C Operational Version:
C
```

4. If an existing program is being modified in any way, the name, date, and version number are required for insertion into the heading coding. An example of a heading after one change:

```
C
C Program Name:
C Written By:
C Date Written:
C Purpose:
C Operational Version:
C
C Changed By:
C Date Changed:
C Reason for Change:
C Operational Version:
C
```

Example of heading after two changes:

```
C
C Program Name:
C Written By:
C Date Written:
C Purpose:
C Operational Version:
C
C Changed By:
C Date Changed:
C Reason for Change:
C Operational Version:
C
C Changed By:
C Date Changed:
C Reason for Change:
C Operational Version:
C
```

5. A record of changes being made to the existing programs will be maintained in detail, to include:

a. The reason for change.

b. Location of change.

6. A record of all files and the authors will be maintained for identification purposes, to include:

a. Author Identification (ID) Number.

b. Device Number.

c. Device ID.

d. Name of File.

e. Length of File.

f. Purpose of File.

D. Object Program Documentation.

1. Changes will be made to the source code and recompiled. No patches will be made to an object program.

2. Only one version of an object program will be kept on file.

E. Summary. Good documentation can be summed up in these four general rules:

1. Provide adequate documentation.

2. Document within program, where possible.

3. Be considerate of the next programmer.

4. Be neat.

III. PROGRAMMING TECHNIQUES

A. Do Loops should have the coding, within the realm of the loop, indented three spaces.

```
Example:  DO 10 N = 1,23
             OUT(N) = 0
             ICT = ICT+N
          10 CONTINUE
```

NOTE: Always use a CONTINUE statement to terminate the realm.

B. Statement Labels.

1. Statement labels will be left justified.

2. Reserve specific ranges of numbers for executable statement labels and format statement labels. Example: Reserve 1 through 999 for executable statements and reserve 1000 and above for format. Labels beginning with 99 would be reserved for termination processing.

C. Variable Name/Constant Between Two Lines of Source Code. Never break up a variable name or a constant between two lines of source code.

D. Statement Continuations. When statements are continued onto a new line, indent the continuation so that similar parts of the statement line up. Example:
```
2000 FORMAT (1HO, 17, F13.4  A1, 10X,
                  17, F14.7, 20X, 13)
```

IV. PROGRAMMING CHECKLIST

A. Program Structure.

1. Is the organization of the program modular?

2. Does each routine and subroutine perform a well-defined, complete function?

3. Is the program's flow straight forward?

4. Does it go from top to bottom instead of jumping around?

B. Dialect.

1. Does your program require assembly routines?

2. Are there assembly routines prepared that meet your needs and do not contain excess functions?

C. Program Development.

1. Have you chosen the best method for the environment in which your program will run?

2. Have you written general and detailed flowcharts?

3. Is your program general and easy to interpret?

4. Can it be used easily with new data?

5. Can it be modified easily?

D. Program Documentation.

1. Has AR 18-7 been followed?

2. Are the program headers present in the source code?

3. Were meaningful comments inserted in the source code?

4. Are all storage locations, that are used, defined?

5. Were the outlined techniques followed?

V. DEBUGGING TECHNIQUES

A. Introduction. There are two basic approaches to debugging:

1. Checking the program before feeding it into the computer.

2. Using the computer to detect errors.

Only after you have exhausted the process of checking your program by hand, should you resort to using the computer to find any remaining errors.

B. Debugging before Compilation or Assembly.

1. After drawing up the flowcharts, put them down for a period of time. Then go back and review them for correctness.

2. Obtain a listing when the coding is finished and verify correctness by desk checking. A few pointers for desk checking:

a. Check that the address part of each instruction refers to the number in memory (the one you wish to make reference to). To aid the checking, you should keep a careful record of where you store data, answers, instructional constants, intermediate answers, etc., while you are writing the program.

b. Check all the loops you have written. Things to check for in your loop:

(1) Is the correct data used in each pass?

(2) Is the computer sent back to the correct place on each pass? In checking this point, you should compare your program carefully with the flowchart.

(3) Will the computer make the desired number of passes through the loop?

(4) Is there an exit from the loop?

(5) Is there a finite limit to the iterations in the loop?

(6) Instruction modification should also be checked carefully in your loop.

c. Whenever a program sends the computer to a subroutine, you should check the calling sequence carefully.

(1) Each subroutine should have a write-up describing the required calling sequence.

(2) Be sure the computer is sent to the correct subroutine.

(3) If it is required that an argument be in the accumulator when the computer exits to a subroutine, make sure you prepare your program to satisfy this requirement.

(4) If the subroutine write-up states that the addresses of the arguments should be included in the calling sequence, be sure these addresses are placed correctly and that they really are the addresses of the arguments, not the arguments themselves.

(5) If the write-up calls for an error return, make certain you have provided one in your program.

d. You can make the program checking process easier by making notes on the coding sheet while you are writing your program.

e. In addition to checking the instructions in your program, make certain the data is written correctly.

f. To make checking of your program easier, you should try to relate the various parts of the program to the corresponding parts of the flowchart.

g. When you have checked your program thoroughly and are absolutely certain it contains no errors; when a second person has gone through your program and found no mistakes; when you are convinced that your program is errorless, then you may test it on the computer to find out whether it works. It is just possible that something may still be wrong with your program.

3. When a new copy or version of the program is made, either date or destroy old copies.

C. Debugging and Testing after Compilation or Assembly.

1. When a new listing of the program is made, either date or destroy old copies.

2. When testing a program, the following principles apply to almost any processing function:

a. Test the normal cases.

b. Test the extremes.

c. Test the exceptions.

D. Summary. We have taken a look at some of the philosophies of program testing and some of the mechanical aids available. If we wanted to boil everything down to four major principles of testing, they would probably be the following:

1. Write the initial program correctly.

2. Think about check-out when coding.

3. Know the debugging tools which are available.

4. Make the program prove that it works.

If we wished to express these principles more concisely, we might state, "Be a meticulous, thoughtful, well informed skeptic." It is as close as you can come to a "magic wand" for program testing, - and it is not too close.

VI REFERENCES

A. Headquarters, Department of the Army, Army Regulation, AR 18-7, Data Processing Installation Management, Procedures and Standards, Chapter 5, paragraphs 2 through g; Appendix J, Chapters 1 through 4.

B. Scott, Theodore G., Computer Programming Techniques, Doubleday and Company, Inc., Garden City, NY, 1964, 1st edition, Chapter VIII.