AN APPROACH TO DEBUGGING

by

CHING-NEU HONG

B. A., Fu-Jen Catholic University, 1969

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

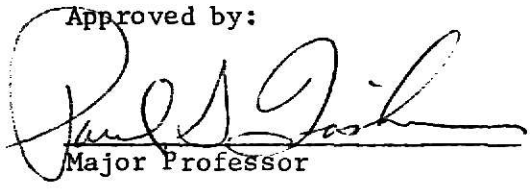
MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1973

Approved by:



Major Professor

TABLE OF CONTENTS

Chapter		Page
I.	INTRODUCTION	1
	What is debugging	1
	Aids to debugging	2
	Objective of the study	6
II.	PROGRAM	7
	Description of the PL/1 program	14
	Limitation and further suggestions	16
III.	SUMMARY	22
IV.	ACKNOWLEDGMENTS	23
V.	BIBLIOGRAPHY	24
VI.	APPENDICES	25
	Appendix A. REPORT program variable list	25
	Appendix B. PL/I program	29
	Appendix C. Original FORTRAN program	36
	Appendix D. Output of the PL/1 program	38
	Appendix E. Output of the modified WATFIV program	40
	Appendix F. Output of the PL/1 program (different data)	43

LIST OF FIGURES

Figure		Page
I.	Flowchart for PL/1 program	19
II.	Flowchart for subroutine program	21

CHAPTER I

INTRODUCTION

I. WHAT IS DEBUGGING?

Debugging is the technique of detecting, diagnosing, and correcting errors which may occur in programs or systems.[1] Debugging may also applied to the process of testing the performance of hardware systems and to the testing of a complete data processing system, but for this paper it relates to detecting, diagnosing, and correcting errors which may occur in programs.

"Syntax errors" and "Logic errors" are two different types of program errors. Syntax errors are errors due to incorrect coding, and logic errors are those due to incorrect appreciation of the problem.

A program written in a symbolic language requires a compilation, or a translation, into the machine language so that it can be understood by the processor. During the process of compilation, syntax errors involving incorrect handling of the symbolic language are detected. Most compilers reject incorrect statements and print out some indication of the type of errors. Normally programs containing compilation errors cannot be run, and the errors must be corrected before the program can be tested. Usually, syntax errors are comparatively easy to find. Here is an example of a syntax error of a DO statement in PL/I. Suppose we

---

[1] Chandor, A., Granham, J. and Williamson, R. A Dictionary of Computers, Australis: Penguin Books, 1970.

write "DO 100 I=1,10". This is a FORTRAN statement, not a PL/I statement. It should be changed to "DO I=1 TO 10;" in the PL/I statement.

An example of a logic error would be to calculate the average score of students. The result is obtained by adding the scores of the students taking the exam and dividing by the number of students taking the exam, not by having the combined total score divided by the number of students in the class. The calculation would be performed correctly by the program, but the result would not be that which was desired. If the program fails to achieve the expected results, or if an unexpected halt occurs within the program, a logic error diagnosis is required. Logic errors can only be detected by testing the program with sample data. The system which will be described in this paper illustrates techniques which could be used in a debugging system to aid the programmer in finding these types of errors.

Frequently, a large part of debugging involves determining what the error is and where it occurred. After finding the error, a correction can be checked by hand or by submitting the corrected program to be run again. In this way, it is possible to determine the essential information which possibly was obscured by the error determination type, or simply by lack of visible information.

II. AIDS TO DEBUGGING

We can say debugging is possibly the most important aspect of programming,[2] as well as the most costly portion of writing a program in

---

[2] IBM Corporation, IBM System 360 Operation System Programmer's Guide to Debugging, IBM Form GC28-6670-3.

terms of both human and machine effort.

When an error occurs in a program, frequently even a very experienced programmer can not identify the cause of the error without considerable effort.

Jacob T. Schwartz in "An Overview of Bugs"[3] discusses debugging and the debugging tools. These tools trace the program at different levels such as the program transfer level, or the subroutine level. Also it is necessary to use different kinds of traps, different kinds of dumps and different kinds of program-termination summaries, such as the number of statements executed in a program run because of the various types of errors which occur in the program. The late-stage debugging tools include a certain number of ideas in the area of systematic program testing. He also mentions a type of debugging aid suggested by Flod's correctness-proof methods.[4] This idea is as follows: one can implement a system by which a programmer, during the process of program elaboration, could formally state those assumptions (concerning the moment-to-moment state of his data) which led him to write his program as he did. However, this method is difficult at best for most programmers.

R. M. Balzer in "EXDAMS-Extendable Debugging and Monitoring System"[5] point out that the ability to debug programs has not progressed as much as the increased use of the higher-level languages has. EXDAMS provides

---

[3] Schwartz, Jocob T. "An Overview of Bugs," Debugging Techniques in Large Systems, Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1971.

[4] ibid. p 10.

[5] Balzer, R. M. "EXDAMS-Extendable Debugging and Monitoring System," AFIPS Conference Proceedings Spring Joint Computer Conference, 1969, pp. 567-580.