AN IMPLEMENTATION OF A FORTRAN SOURCE CODE REARRANGER AND DOCUMENTATION GENERATOR PROGRAM

by

ARTHUR SCOTT MYERS

B.S., Lehigh University, 1969

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements of the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1983

Approved by:

Major Professor
# TABLE OF CONTENTS

I. INTRODUCTION .........................................................1

II. DESIGN REQUIREMENTS ..............................................6

III. DESCRIPTION OF IMPLEMENTATION .................................12

IV. FUTURE WORKS AND EXTENSIONS ..................................35

V. CONCLUSIONS ..........................................................39

APPENDIX A - PROGRAM SOURCE CODE LISTING ......................43

APPENDIX B - USER'S MANUAL ..........................................44

APPENDIX C - EXAMPLES OF ANALYZED PROGRAMS ..................48

BIBLIOGRAPHY ............................................................49
I. INTRODUCTION

One of the most important, and unfortunately most neglected needs in the software development process today is documentation, especially documentation that aids the programmer or systems analyst in readily understanding computer programs' source code. The intent of this implementation project is to address that need by developing a software tool that will provide documentation showing a program's hierarchical top-down order and alphabetical listing of its modules; it will also rearrange the program's source code into this top-down order of modules.

This documentation should have a very high utility to the programmer because it will concisely show, typically on one or two pages, a complete program's overall structure and detailed inter-module relationships via the report that shows the top-down order of modules. This could be viewed as a table of contents for the program's source code, because it will show each module's location within the source code file. It should be even more useful since the same report will also contain structure, size and inter-module relationship information for each module. The report displaying the alphabetical listing of modules can be thought of as an index to the program, but it too will contain information that goes beyond that function. Furthermore, since the source code file will actually be rearranged into its hierarchical top-down order, an implicit documentation feature will come about because when one reads the source
code it will be in logically related order. Thus, if the reader wants to look at the code of a module called by the module he or she is currently studying, it will usually be a matter of flipping to the next page or two, in most cases, instead of having to make a time consuming search through the entire source code listing.

Because of the documentation and organization capabilities that this software tool will provide, it should prove to be quite valuable to the programmer enabling he or she to improve both productivity and quality with the following benefits:

(1) ordering of source code in top-down order
(2) program documentation shown in top-down and alphabetical order
(3) understanding of program structure
(4) rudimentary measure of software quality - e.g. program unit sizes, percent of comments, etc.
(5) improved productivity in developing new programs
(6) greater ease of maintaining and enhancing existing programs

While using this tool in developing and maintaining one's own software should be of considerable worth, it should be even more useful when one has to work with programs developed by someone else. Furthermore, the rudimentary measures of software quality provided should be of value to the software manager.

Previous work that has been done in this area can be
classified into two approaches: (1) analyzing module relationships of FORTRAN programs which centers around inter-module communications and how the data shared between the modules are affected and (2) identifying basic blocks of FORTRAN programs. The first approach constructs the call graph of a FORTRAN program [2] in order to examine interprocedural communications. This call graph serves as a data structure for providing static representation of interprocedural relations for nonrecursive programs. The algorithm for constructing the call graph is used in the PPORT Verifier [3], a program which checks FORTRAN programs for compliance with a portable subset of FORTRAN. The uses of the call graph in the Verifier are:

(1) checking the legality of module linkage (i.e. the matching of parameter number, usage, type, and structure)
(2) checking for unsafe references to interprocedural communications which may have implementation dependent results
(3) checking the sharing of global data areas
(4) outputting a depiction of the call graph for use as documentation and as a debugging aid

The construction of a program's call graph differs from this project because it does not actually rearrange the source code, nor does it create the tabular documentation showing the program's top-down order (table of contents) and its alphabetical listing of modules (index). On the other hand,
this project does no checking of parameters passed between modules or detecting of non-portable features in the program.

In the second approach, a FORTRAN program was written to identify basic blocks in FORTRAN programs [1] where a basic block is defined as a sequence of consecutive FORTRAN statements which must be executed consecutively; that is, if one statement in the block is executed, all are executed. Use of this program allows for monitoring of the execution sequence and control path analysis of a program. This approach differs from the source code rearranger about to be described because it must perform a rather complex analysis within modules in order to determine the boundaries of the basic blocks, whereas the source code rearranger is concerned about only module boundaries and any inter-module calls that may reside there. And, as with the first approach, no rearranging of source code and related documentation is obtained via this basic block identifier.

Both of these approaches address related but lower level concerns than does this implementation project. The output from the code rearranger program will be easily understandable to any programmer and possibly non-programmers and will have the primary value of aiding the programmer in developing, maintaining, and enhancing software. Thus this software tool will have legitimate use on a day-to-day basis. In the approach that constructs the call graph, considerable effort was directed toward formalizing the algo-
rithm and proving its correctness. In the identifying basic blocks approach, great care was taken to adhere to the definition of a basic block and to gracefully handle the exceptional conditions that occur. Thus, one could also view the emphasis of this project as being more practical and less theoretical than the two approaches just described.
II. DESIGN REQUIREMENTS

In considering design requirements for this implementation project, it was decided that this code rearranger program should be able to process both FORTRAN 66 and FORTRAN 77 programs and be capable of handling fairly large programs (up to 4000 lines of source code) that are developed and run in a mini-computer environment. Recognizing that both FORTRAN 66 and FORTRAN 77 are in use today, a higher utility of this program could be achieved by being able to process both versions' source code files. Being able to analyze programs with up to 4000 lines of source code would allow for this program to serve as a software tool for much of the FORTRAN software being written on mini-computers today. With the increasing power, and therefore usage of mini-computers, brought about primarily because of technology gains in hardware, this FORTRAN source code rearranger program could have a large number of potential users, assuming, of course, that it would be portable from one machine to another. Thus, the above considerations were made with the objective of making the results of this project as widely available and useful as possible.

In order to prevent the design of this project from becoming overly complex, it would be clearly stated to any users that satisfactory performance of this program could be guaranteed only if it was used AFTER successful compilation of the FORTRAN program that was about to be analyzed and rearranged. Obviously, if this program had to embody the
capability to detect syntax and semantic errors pertaining to module structure, it would be much more complex than if the resident FORTRAN compiler processed programs prior to the rearranger program doing its analysis. Thus, with the stipulation that the code rearranger program would process only "clean" FORTRAN source code, designing of this post compiler tool became a very doable project.

The program would detect errors that are typically discovered by the linking-loading process, but it would be good and efficient practice to use this program AFTER the linking-loading of the FORTRAN program had successfully been done. By successfully performing the linking-loading step prior to using this program, the most correct program possible would be passed to the source code rearranger program. Thus, the highest probability of successfully rearranging the program on the first attempt would be assured.

Next, it was determined what general tasks this program would have to perform so that it would be a desirable and valuable aid to the programmer. The three general tasks selected were:

1. rearrange the program's source code file into hierarchical top-down order of modules. This rearranging would be successful regardless of the order of the modules in the original source code file as long as the original source code file was complete, i.e. contained all of the program's modules.

2. generate documentation showing the top-down order of
program modules. This report includes, for each module, the beginning and ending source code line numbers; the level of its call ("main" is defined as level 1); the number of lines of code; the number of comment lines; the number of declaration lines; the number of format lines; and the number of executable lines. Also, this report would print out totals for these module statistics. Thus, this report serves as a table of contents to the now hierarchically ordered source code file.

(3) generate documentation showing the alphabetical listing of the program's modules. This second report shows, for each module, the beginning and ending line numbers; the number of lines; and the number of times called. This report would also display the range and average of modules' size and percent of total lines of code by comment, declaration, format, and executable. This second report would become the index to the rearranged source code file.

With these three outputs of the source code rearranger program: (1) the rearranged source code, (2) the table of contents, and (3) the index, the programmer has a very useful documentation package which has mapped the entire source code's logical structure to its physical structure while also allowing for quickly determining specific module information and relationships. This should prove to be a valuable tool for developing, maintaining, and enhancing FORTRAN
software.

After defining the general tasks, a list of specific considerations was developed that this program would have to be capable of handling. Taking into account the expected environment of the FORTRAN programmer, the following requirements were formulated:

(1) In striving toward the goal of having a useful and practical tool for aiding the FORTRAN programmer, the design requirement of processing FUNCTION modules in addition to MAIN and SUBROUTINE modules was deemed important and necessary. The code rearranger program would give functions the same precedence as subroutine modules in constructing a program’s calling hierarchy but would identify each module as "MAIN", "SUBROUTINE", or "FUNCTION" in the subsequently outputted documentation.

(2) Recognizing that virtually all FORTRAN programs contain library calls (calls to modules that exist in the system library, not in the program’s source code), it was determined that this program must be able to detect such module calls and process them appropriately. Any call in the source code referencing a module not found in the source code would be assumed to be a library call and hence would simply be ignored when building the module hierarchy.

(3) Being aware that there are now versions of FORTRAN, e.g. HP-3000, that allow for recursive routines (modules
that call themselves), the requirement to detect and properly document recursion became necessary. If recursive calls were not detected and so labeled, then the report showing the hierarchical relationship of modules would grow to infinite length.

(4) A common error that can easily occur when developing FORTRAN programs of even moderate size is duplicate naming of modules, which will not be properly detected by the compiling and linking-loading processes. Because of this fact, they can be especially troublesome to the programmer as he attempts to debug his program. By being able to detect duplicate module names, this program not only assures the integrity of the ultimately rearranged program, but also performs a valuable debugging service.

(5) Another common error that happens when writing FORTRAN software is having extraneous modules in the source code - those which are never called by any other program module. This error, while not fatal, does undermine good housekeeping and space conservation practices. The code rearranger program will naturally expose any extraneous modules as it does its hierarchy analysis.

(6) Because FORTRAN allows for EXTERNAL modules, those passed as arguments in one module, but defined in another module, the capability for proper analysis of the condition must exist in this program. Without this ca-
pability, an incorrect analysis of the hierarchy of module relationships would result.

(7) Finally, keeping in mind that this program must be able to process moderately large programs (up to 4000 lines of code), the determination as to how to handle data storage (arrays in CPU and/or disk files) during program execution was made so as to optimize program execution time. A satisfactory compromise was to handle the input and output source code as a disk file while all other data requirements were to be processed using arrays residing in CPU.
III. DESCRIPTION OF IMPLEMENTATION

In order to generate the source code file in top-down module order and associated documentation, the computer program that was implemented must create from the input source code file (whose modules can be in any sequence) a table that includes all the module boundaries with all module calls in the proper order within these boundaries. Constructing the table may require up to three passes through the source code, if FUNCTION(S) and/or EXTERNAL subroutine(s) are present. When the table is complete, the rearranged source code file can be output by traversing through this table utilizing a stack in order to determine the hierarchical relationships of the modules. The hierarchy diagrams of Figures 1-8 show an overview of the program design. We will now present a thorough explanation of how this program accomplishes the rearranging of modules and the related report generation:

(1) A first read of the entire source code file is performed so that a 'reduced' array containing only program unit statements (e.g. SUBROUTINE OR FUNCTION) and CALL statements is built. During this first pass, using subroutines FINDSEGBEGIN and CLASSIFYSTATEMT to analyze each program unit, every statement is classified as comment, declarative, format, or executable and the beginning and ending line numbers are determined. This information is also stored in the 'reduced' array. Also, for each program unit name appearing in a program unit
LEVEL OF CALL

1

2

DEFINEFILES  FINDSECBEGIN  CLASSIFYSTATEMT  DELETELIBCALLS  INSERTFUNCALLS  COMPLETEARGLIST  FINDSAME-NAMES  DETERMINIF-CALLED  REARRANG-SOURCE  COMPUTSTAT-ISTIC

3

SEE FIGURE 2  SEE FIGURE 3  DETERMINE-NAME  SEE FIGURE 4  SEE FIGURE 5  SEE FIGURE 6  SEE FIGURE 7  SEE FIGURE 8

FIGURE 1. HIERARCHY DIAGRAM OF SOURCE CODE REARRANGER PROGRAM

NOTE: MODULES ARE EXECUTED IN VISIT-LEFT-RIGHT SEQUENCE (PREORDER)
FIGURE 2. HIERARCHY DIAGRAM (CONTINUED)
LEVEL OF CALL

FIGURE 3. HIERARCHY DIAGRAM (CONTINUED)
FIGURE 4. HIERARCHY DIAGRAM (CONTINUED)
statement, an entry is made to a 'name' array.

(2) Next, via the subroutine DELETELIBCALLS, any library calls are determined by checking for the existence in the 'name' array of each name found in CALL statements. If the name is not found, then the call is assumed to be to a system library routine and its CALL statement is deleted from the 'reduced' array.

(3) If there are any FUNCTION program units in the source code file, a second read of the file takes place in order to find the function calls (since functions are not called by the explicit CALL statement). These function calls are then inserted into the 'reduced' array. The subroutine INSERTFUNTCALLS performs this task.

(4) A third pass of the source code file becomes necessary if there are any 'EXTERNAL' subroutines. In the third pass, using the subroutine COMPLETEARGLIST, a table is completed that links the actual and formal parameter names of the EXTERNAL subroutines.

(5) At this point the subroutine FINDSAMENAMES detects any duplicate program unit names. If any are found, the name(s) and locations in the unchanged source code file are shown and the program stops.

(6) Next, the subroutine DETERMINIFCALLED checks for the existence of program unit(s) that are not called by at least one other program unit. If any are found, their
name(s) and location(s) in the unchanged source code file are shown and the program stops.

(7) The source code file is now rearranged by the subroutine `REARRANGESOURCE` into its hierarchical top-down order of program units. Concurrent with the source code file being rearranged, the first report showing this rearranged order is output.

(8) Lastly, the second report, showing the alphabetical listing of program units with location and calling frequency and statistical summary, is output via the subroutine `COMPUTSTATISTIC`. This causes a normal termination to execution of the program.

The above discussion should give the reader a basic understanding of the purpose and method of this code rearranger program. The narrative about to follow will give a more detailed explanation.

We will now give a brief description of each program module of this code rearranger program, once again referring to the hierarchy diagrams of Figures 1-8:

**MODULE DESCRIPTIONS**

**MAIN** - Naturally, the program begins with this program module. It declares the necessary global variables and controls the sequential execution of the subroutines that perform the rearranging of the source code file and associated report generation. By looking at the source code and embedded comments of MAIN, the reader should be able to quickly understand, at a very high level, the steps taken by this program to perform its functions.
Subroutine DEFINEFILES (called by MAIN) — This module queries the user for the name of the source code file to be processed so that it can define and initialize the needed input and output disk files for storing the source code.

Subroutine FINDSEGBEGIN (called by MAIN) — As its name implies, this subroutine finds the first line (ignoring comment lines) of the current program module of the input source code file. The current module is determined by reading the next record of the source code file. (The current record location is passed to this subroutine by MAIN.) It then determines if this program module is MAIN, a function, or a subroutine and stores this information along with the module name and beginning line number in the REDUCED array. If the program unit is a function or subroutine, the program unit name is added to the appropriate name array (FUNCTIONNAME or SUBNAME).

Subroutine STRIPOFFPARAM (called by FINDSEGBEGIN and PROCESSCALL) — This module receives as input either a program unit statement or a CALL statement and removes the argument list, including parentheses, from the statement. It then returns this shortened string to the calling subroutine for eventual entry into the REDUCED array.

Subroutine STORENAME (called by FINDSEGBEGIN) — This short procedure parses out the program unit name from the program unit statement and enters it into the name array passed to it by FINDSEGBEGIN.

Subroutine DETERMINENAME (called by STORENAME, CHECKARGLIST, DELETELIBCALLS, and DETERPROGCALL) — This module, passed a character string and a beginning location in that string, returns a sixteen character variable NAME.

Subroutine CLASSIFYSTATEMT (called by MAIN) — This program unit classifies each statement of the current program unit as comment, declaration, format, or executable and increments the appropriate counter. If it encounters any CALL statements, they are entered into the REDUCED array after the necessary processing. If an EXTERNAL statement exists in the current program unit, then the CALL statement is processed as required to link any external arguments of the CALL statement to their location in the argument list. When the END statement
of the program unit is reached, the ending line number and statement type counts are stored in the REDUCED array and program control is returned to MAIN.

Subroutine DETERDISORFMT (called by CLASSIFYSTATEMT and INSERTFUNCTCALLS) — This subroutine determines if the current source code statement is of type DISPLAY or FORMAT and increments the corresponding counter accordingly.

Subroutine PROCESSCALL (called by CLASSIFYSTATEMT) — This module performs the necessary steps for making the CALL statement suitable for entry into the REDUCED array.

Subroutine PROCESSIF (called by PROCESSCALL) — This subroutine, passed a CALL statement contained in an IF statement of more than one line, removes the argument of the IF statement and thus returns a string that guarantees the presence of the program unit name that follows 'CALL'.

Subroutine CHECKARGLIST (called by CLASSIFYSTATEMT) — This program unit, passed a CALL statement of a program unit that has an EXTERNAL declaration, determines if any arguments of this CALL statement are declared in the EXTERNAL statement. If any are, the following information is stored in the ARGSUBSTITUTION array:

1. subroutine name called by this CALL statement
2. name(s) in CALL statement's argument list found in EXTERNAL statement
3. position number(s) in CALL statement's argument list of argument name(s) found

Then, the CALL statement stored in the REDUCED array is marked with the ARGSUBSTITUTION array position where the above information has just been stored.

Subroutine DETERDECLARE (called by CLASSIFYSTATEMT and INSERTFUNCTCALLS) — This module determines if the statement passed to it is declarative, and if so, increments the corresponding counter. Also, if the statement is of type EXTERNAL, it turns on a flag signifying such.

Subroutine DELETELIBCALLS (called by MAIN) — This module deletes system library calls from the REDUCED array. It does this by checking for the presence in the SUBNAME array of the subroutine name referenced in each CALL statement. If it does not find the name in the SUBNAME
array, then that CALL is assumed to be a system library call and is removed from the REDUCED array.

Subroutine INSERTFUNCTCALLS (called by MAIN) - This subroutine is called if one or more FUNCTION modules are present in the input source code file. Its purpose is to read through the entire source code file in order to find FUNCTION calls and their locations and add them to the REDUCED array. To locate FUNCTION calls, this subroutine looks for the presence of any FUNCTION name (stored in array FUNCTIONNAME) in each source code statement (bypassing comment, declarative, and format statements). When a FUNCTION name is located, it insures that it is not its own FUNCTION declaration, and then enters the FUNCTION name and line number location prefixed by 'CALL' into the REDUCED array. After it has processed the last line of source code, it then sorts the REDUCED array by line number location so that all program unit and CALL statements are in the same sequence as the input source code file.

Subroutine SORTARRAY (called by INSERTFUNCTCALLS, COMPLETEARGLIST, and LISTNAMES) - This program unit sorts the REDUCED array into the order specified by the values of KEYSTART and KEYLENGTH that are passed to it by the calling subroutine.

Subroutine SWITCH (called by SORTARRAY) - This subroutine simply switches the location of two adjacent records passed to it by the subroutine SORTARRAY.

Subroutine COMPLETEARGLIST (called by MAIN) - This module is called if one or more EXTERNAL statements exist in the input source code file. Its task is to link the actual parameter module name(s) (referenced in the EXTERNAL and CALL statement(s) of a program unit) with the formal parameter name(s) of the module(s) called by the CALL statement(s). To do this, for each module that contains an EXTERNAL statement, the CALL statement that is passing the EXTERNAL parameter name(s) is found in the REDUCED array. This entry contains the pointer to the ARGSUBSTITUTION array (established by subroutine CHECKARGLIST) where the argument name (actual parameter) and its list position in the CALL statement are read. The location of the module called by the CALL statement is found in the REDUCED array. Thus, the argument name of the same list position (formal parameter) of the module statement, determined by reading the correspond-
ing line in the source code file, is added to the ARGSUBSTITUTION array. Then, the CALL statement using this argument is found in the input source code. This CALL statement, with its source code location and the notation 'PASSED VIA EXTERNAL', is added to the REDUCED file. When all necessary modules have been processed, the REDUCED array is once again sorted by ascending line number location to preserve this array's consistency with the input source code file.

Subroutine GETSUBARGNAME (called by COMPLETEARGLIST) - This subroutine finds the subroutine statement in the input source code for the subroutine referenced by each CALL statement that is passing EXTERNAL parameter(s). Then, it parses out the argument name based on the list position passed to it and links this formal parameter name with the actual parameter name in the ARGSUBSTITUTION array. Lastly, it finds the location of the CALL statement in the source code file that references this formal parameter name.

Subroutine FINDSAMEARRAY (called by MAIN) - This program unit determines if any modules are sharing the same name. It does this by merging and sorting the FUNCTIONNAME and SUBNAME arrays. Then, if it finds any two adjacent names to be identical, it displays this information and aborts the program.

Subroutine SWITCH (called by FINDSAMEARRAY) - Just like SWITCH, this module switches the location of two adjacent records.

Subroutine DISPLAYDUPARRAY (called by FINDSAMEARRAY) - This subroutine gets the input source code's beginning and ending line numbers of duplicately named modules from the REDUCED array and displays the module name and the beginning and ending line numbers for each occurrence of the module name.

Subroutine DETERMINECALLED (called by MAIN) - This module checks for any program units that exist in the source code but are never called. It does this by finding each CALL statement in the REDUCED array and incrementing the counter for the corresponding function or subroutine name. If any counter still has a value of zero after completion of this process, then the module name(s) and beginning and ending line numbers are displayed and the program is aborted.
Subroutine DISPLAYNOCALLS (called by DETERMNIFCALLED) — This program unit displays the name(s) and location(s) of any module(s) present in the source code but never called. Before the display, it checks to see if the module is referenced via an EXTERNAL call by checking the ARGSUBSTITUTION array.

Subroutine REARRANGESOURCE (called by MAIN) — This module rearranges the input source code file into its hierarchical top-down order of modules, thereby creating an output source code file. Concurrent with this rearranging of modules, a report showing this rearranged structure is generated. This report shows for each program unit name, new beginning and ending line numbers; level of its call (MAIN is defined as level 1); number of lines of comment, declaration, format, and executable lines of code. Also, appropriate source code totals are shown. This module accomplishes its task by using the information contained in the REDUCED array to select out the modules from the input source code file in the proper sequence. The first module of the rearranged source code is always MAIN. With this starting point, traversing through the REDUCED array and employing a stack to keep track of return addresses within the array, the hierarchical rearranging of modules takes place. When rearranging is complete, this module checks that the number of lines of the input source code file equals the number of lines of the just created rearranged output source code file. If such is the case, then the output file overwrites the input file. If this condition is not met, the appropriate message is displayed, and the program is aborted.

Subroutine PAGEHEAD (called by REARRANGESOURCE and PRINTEXT) — This subroutine prints out the title and column headings for the report that shows the rearranged source code information.

Subroutine FINDPROGRAMUNIT (called by REARRANGESOURCE and DETERPROGUCALL) — This program unit finds the program unit statement in the REDUCED array for the program unit name contained in the string passed to it. Also, if the passed string includes 'RECURSIVE CALL' in it, then a flag denoting recursion is turned on.

Subroutine INDENTSUBNAME (called by REARRANGESOURCE) — This module appends zero or more leading blanks to the string passed to it. The number of blanks is determined by the value of the stack pointer which represents one less than the level of the call for the current program
unit about to be displayed.

Subroutine PRINTLINE (called by REARRANGESOURCE) - This subroutine prints the line for the current program unit. If a line for the program unit has been printed previously, only the name and level of the call is output; otherwise, the complete line is printed. A counter is incremented which keeps track of how many time this program unit has been called, line totals are incremented, and the new beginning and ending line numbers, number of lines, and number of times called are entered into the REDUCED array.

Subroutine ACCUMLINETOTALS (called by PRINTLINE) - This program unit simply increments the counters for report totals by the amounts for the current program unit.

Subroutine WRIETETONEWFILE (called by REARRANGESOURCE) - This module writes the lines of the current program unit to the rearranged output source code file.

Subroutine DETERPROGUCALL (called by REARRANGESOURCE) - This subroutine is passed the string that contains the current CALL statement and determines if it is a recursive call or if name substitution is required because of passing the program unit name via an EXTERNAL statement. If name substitution is required, the ARGSUBSTITUTION array is accessed as required.

Subroutine PRINTTOTALS (called by REARRANGESOURCE) - This program unit prints the totals for the report that shows the hierarchical top-down order of modules and associated information.

Subroutine SAVENEWFILE (called by REARRANGESOURCE) - This subroutine overwrites the input source code file with the rearranged output source code file after it has determined that the number of lines in each file are the same. If this is not the case, then the appropriate message is displayed and the program is aborted.

Subroutine COMPUTSTATISTIC (called by MAIN) - This module outputs a report showing the alphabetical listing of the program unit names of the source code file. Included in this report are beginning and ending line number and number of times called for each program unit. Also displayed are the largest, smallest, and average
program unit size, and percent of lines by type (comment, declaration, format, and executable).

Subroutine LISTNAMES (called by COMPUTSTATISTIC) - This subroutine extracts the program unit statements from the REDUCED array which contains all the necessary data for outputting the alphabetical listing report. It then sorts this list of program unit statements into alphabetical order by program unit name. After printing the page heading, it then displays this alphabetical listing.

Subroutine PAGE2HEAD (called by LISTNAMES and COMPUTSTATISTIC) - This module prints out the title and column headings for the report that shows the alphabetical listing of source code modules.

OTHER IMPLEMENTATION ISSUES

Analysis shows that this implementation meets all the design requirements stated in Section II. This code rearranger program can accommodate both FORTRAN 66 and FORTRAN 77 source code files of up to 4000 lines. When used as a post-compiler, post linkier-loader, this program will perform the three general tasks stated in the Design Requirements: rearranging of the input program's source code file into hierarchical top-down order of modules; generating of documentation showing this top-down order; and generating of the report showing the alphabetical listing of the program's modules. The specific tasks listed in Section II are also met: FUNCTION modules are processed appropriately; library calls are detected and eliminated from the module hierarchy; recursive routines are discovered and handled efficiently; duplicately named modules are found and this error condition duly noted; likewise, modules present in the source code but
never called are caught and displayed; EXTERNAL modules are correctly analyzed by constructing a table that relates the actual parameter names with formal parameter names of passed subroutines; and data storage is handled wisely by using disk files for the input and output source code and arrays for all other data requirements.

Extensive testing was done in order to determine that this program would perform as stated in the Design Requirements. A practical approach was taken here—that of processing many programs many times with this code rearranger program. Well over one hundred different source code files (ranging in size from fifty lines to 3,700 lines) were used for ascertaining that this software could perform its designed function in a robust and correct manner. This high volume of testing activity served very effectively to expose programming errors, which were corrected as they were encountered.

Using the prototyping methodology was of significant benefit in implementing this project. This approach to software development, analogous to designing prototypes in engineering applications, allows for quick feedback and early detection and correction of design errors. In fact, using this method was the main reason that extensive testing could be done during all of the software development process. Referring again to the hierarchy diagram of Figure 1, one can see that this program's design is compatible with the iterative testing inherent in prototyping. For example,
it was possible to correctly process programs containing no functions before the code needed for processing of functions (primarily subroutine INSERFUNCTCALLS) was ever written. Likewise, programs containing no EXTERNAL modules were successfully rearranged prior to the existence of any of the code pertaining to EXTERNAL modules (primarily subroutines CHECKARGLIST, COMPLETEARGLIST, and GETSUBARGNAME). Thus, in a matter of only a few weeks, a version of this program was created that could properly rearrange 'simple' FORTRAN programs and validate that the essential software for rearranging source code into its hierarchical top-down order performed correctly. Proceeding from this sound base, extending the program to accommodate all of the remaining design considerations became an orderly and efficient task.

The issue of portability was addressed with this implementation project as this program was required to run on a Hewlett-Packard HP-3000 computer and on a Western Electric 3B20 computer (under the UNIX operating system). The code rearranger program was originally written for the HP-3000 computer whose version of the language is FORTRAN 66 with enhancements added by Hewlett-Packard, such as variable names of up to 15 characters, character variables including substring manipulation, and ACCEPT and DISPLAY statements for processing unformatted input/output. Making this program compatible with the FORTRAN 77 running under UNIX on the 3B20 machine necessitated shortening all variable names to six or less characters, replacing ACCEPT and DISPLAY
statement with READ and WRITE statements, modifying some language dependent features such as file access statements and handling of mixed mode arithmetic, and revamping character declaration and substring manipulation statements because of syntactic and addressing differences. Thus, the source code for the two machines could not be identical because of some of these fundamental language divergencies. But, except for the syntax and character substring differences, only two subroutines required significant modification to any of its statements - DEFINEFILES and SAVENEWFILE - both because of file access statements. Although, true portability of source code was not achieved in this case, the differences were reduced to the absolute minimum.

This program should be able to run on other computer systems that support FORTRAN 66 or FORTRAN 77 without requiring extensive modification to its source code. Any machine running under UNIX should be able to accept the FORTRAN 77 version of the source code with little or no modification. In fact, a considerable effort was made to port this program to KSU's Perkin Elmers Interdata 8/32 computer which runs under UNIX. The only reason portability was not achieved was because of a serious bug in the FORTRAN 77 compiler. When the compiler has been debugged, the porting of the 3B20's source code to the 8/32 should be a straightforward exercise. For other computers, depending on their machine dependent features and customizations to FORTRAN, adaptations probably will be necessary to one or the other
source code files. But since two versions of the source code do exist, one of them should be virtually compatible for almost any mini-computer. Unfortunately, complete portability is not possible at present because of lack of standardization of software and hardware between computer vendors.

As with any implementation project, there are limitations and constraints for this one which are primarily determined by the computer environment in which this program is designed to run. Specifically, as the code rearranger program is presently written, the following maximum capacities apply to the program that is to be rearranged:

1. 4000 lines of source code
2. 100 subroutines and 100 functions; 150 functions and subroutines combined
3. 370 (CALL + SUBROUTINE + FUNCTION) statements

These upper limits were dictated by the allowable maximum run-time data stack of the machine (HP-3000) where the program was developed. Obviously, if the program is transported to another computer environment, these limits should be reviewed and adjusted accordingly. An alternative for increasing the size of programs that can be processed without changing the computer environment would be to modify the program to use disk files in place of some or all of the arrays that are now specified, thereby reducing the required data stack. The user should be aware, however, that such a change will increase the required execution time of the pro-
gram because of higher demands for disk input/output. In any case, this program should be able to be customized to the degree necessary to satisfy the specific user's requirements.
IV. FUTURE WORK AND EXTENSIONS

Obviously, this project has opportunities for future work that will enhance its value to the programmer or systems analyst. Naturally, extending this application to process programs of other languages, e.g. PASCAL, C, COBOL, etc. would be both worthwhile and feasible. In the case of PASCAL, considerations relating to forward referencing of procedures would have to be taken into account. Forward references would have to be inserted into the source code, if they did not already exist, so that the hierarchical top-down module order of the rearranged file would compile successfully. Furthermore, these forward references would have to be treated as comment statements belonging to no program unit when the source code rearranging takes place. Otherwise, duplicate module names would be detected and the rearranging process would abort. For the C and COBOL languages, rearranging of code should be as straightforward as for FORTRAN. In any case, two important decisions to be made before extending this project to another language are:

1. What version(s) of the selected language would the developed software support?

2. In what version of what language would this source code rearranger software be written?

The answers to these two questions will promulgate specific design considerations and also very much determine portability.
Another approach to extending this system to other programming languages might be writing the source code rearranger program to be table driven. Depending on what language is being processed, the corresponding table containing keywords for identifying call and program unit statements would be accessed. These tables would also contain keywords for enabling classification of statements by comment, declarative, executable, etc. A difficult issue that would arise here would be detecting of comment lines in C, for example, where the delimiters for the comment ('/*','*/') can appear anywhere on the line. This certainly would have to be handled differently than in FORTRAN where the comment symbol ('C') must always appear in column 1. Careful analysis of each programming language's calling structure would be necessary so that the hierarchical relationship between modules could be established. Fortunately, this project has already considered both explicit (subroutines) and implicit (functions) types of calls. Thus, it certainly should be practical to extend this approach to other languages such as PASCAL and C which have implicit procedure and function calls. One more problem area for implementing this 'table driven' approach is how to process languages, such as PASCAL and C, which allow multiple statements per line. In order to establish a useful relationship between number of lines and number of statements, as well as to define what parsing capabilities would be needed, ground rules would have to be agreed upon that would standardize
how this situation is to be handled. But the capability of one program to handle a variety of languages would make grappling with issues like the above very justifiable.

Another very useful feature of this project would be the capability to generate graphical outputs, in addition to the report formatted documentation, that would be essentially of the form of the hierarchy diagram shown in Section III. Implementation of this feature would be primarily dependent on defining standard graphical notation. Also, handling such situations as depicting modules whose call invocations are at more than one level and scaling of the graph based on the number of modules, calls, and levels would pose a challenge. Portability would become a difficult issue as another piece of hardware, the graphics output device, would be needed for implementation. But the benefit of having a complete program's structure succinctly displayed in graphical form would certainly justify the development effort.

An additional enhancement to this project would be the incorporation of a data-base, interfaced with this code rearranger program, that would maintain program/module relationships, thereby providing the programmer with valuable information. If this data-base had the capability to catalog detailed information about all of a system's programs' module structure, such as module name, line counts by statement type, number of characters, modification dates, programs used by, etc., then the user would have a powerful
tool for maintaining and controlling software development of his or her source code. For example, if a programmer has to change the code of a specific module, by accessing this data-base, he or she could readily determine what programs are affected and therefore have to be maintained. This could be especially valuable if one is responsible for hundreds of programs, many of which use common subroutines or functions. Also, the user could query this same data-base to ascertain what program units a specific program is comprised of without having to actually look at the program's source code. It might even become possible to create a new program by simply specifying the names and locations of existing routines. An important consideration in implementing this feature is selection of the data-base management system where this data will reside. This choice will greatly influence portability to other computers. As with the other suggested extensions, it appears that the most difficult to achieve software engineering goal will be a high degree of portability. Hopefully, in the years to come, as computer science becomes more of a science, and less of an art, software and hardware incompatibilities will lessen so that portability will be a much more easily realized aim.
V. CONCLUSIONS

In conclusion, the question should be asked: What are this project's benefits and do these benefits meet the needs and desires stated in the introduction to this paper? To address this question, we will restate the six benefits listed in the Introduction as questions and supply the answers:

(1) Is source code rearranged into top-down module order? Yes. This has clearly been accomplished, as the reader can see by referring to the example programs of Appendix C.

(2) Is program documentation showing top-down and alphabetical module order generated? Yes. Please refer to Appendix C once again and examine the two reports entitled "SUMMARY OF EXECUTION SEQUENCE OF PROGRAM UNITS FOR FORTRAN SOURCE CODE: ..." and "LISTING OF PROGRAM UNITS IN ALPHABETICAL ORDER WITH STATISTICAL SUMMARY FOR SOURCE CODE: ..."

(3) Does the rearranged source code and associated documentation give a better understanding of the program's structure. Yes, in two ways: obviously, having the complete program's top-down module structure contained on a one or two page report lends itself to greatly improved user understanding (at a high level), and the alphabetical listing of modules makes possible quickly locating of a module's source code; but in a more subtle but no
less important way, the mere fact that the program's source code is arranged in the same order as the program's execution sequence, makes it much easier for the programmer to gain a detailed understanding of the program's instructions as he or she follows the flow of control through part or all of the source code listing.

(4) Does the output of the code rearranger program provide a rudimentary measure of software quality? Yes. Referring to the alphabetical listing of program modules for one of the sample programs in Appendix C, one can readily see the largest, smallest, and average program unit sizes as shown at the bottom of this report. Also, the percent of comment lines contained in the source code is prominently displayed. Simple, but important statistics such as these can readily indicate if the program employs modular structure and whether it is adequately commented. Furthermore, the column of this report that shows how many times each module is called could prove worthwhile in analyzing and improving a program's execution time as is would highlight any routines that are called numerous times.

(5) Can one expect improved productivity in developing new programs? Yes, especially if the programmer writes modular code in conjunction with using the prototyping methodology that was discussed in Section III. Then, with each iteration of software development, the pro-
grammer has up to date documentation and properly ar-
ranged source code of the in- development program. 
These tools will naturally lead to more efficient and 
higher quality software development practices.

(6) Will maintenance and enhancements of existing pro-
grams be made easier? Most emphatically, yes! When the 
occasion arises for a programmer to refer to existing 
source code, which could be several years old and writ-
ten by someone else, the fact that this code is in 
hierarchical top-down order, with the associated 'table 
of contents' and 'index' reports, will prove to be in-
valuable for providing quick and in depth understanding 
of the program's intended function. Thus, the 
programmer's time and talent will be much more effec-
tively used because he or she will almost immediately be 
able to start work on developing the necessary software 
changes or additions instead of having to spend hours or 
days studying and manually rearranging the source code 
listing so that it can be understood.

Summarizing, we were able to answer affirmatively that all 
six benefits stated in the Introduction were achieved with 
the implementation of this project. Thus, one can deduce 
that this project began from a well defined problem state-
ment and ended with a successful problem solution.

One aspect of implementing this project that was unex-
pected was the firsthand realization of how very powerful 
software tools can be. This power was most strongly felt
during the implementation phase of developing this software. Simply stated, this program aided in its own creation and evolution. As discussed in the previous paragraph, because this tool can improve programmer productivity, especially when coupled with the prototyping approach, the speed and efficiency with which this code developed was greatly attributed to the ability of this program to analyze and rearrange itself. This early on functional capability existed, because of prototyping, long before many of the modules were written (see Section III). Thus, this recursive elegance accelerated each successive iteration of code. Also, the power of this software tool was noticed in using it on a daily basis for its designed function of processing other programs. Being able to perform the rearranging and documentation for a several thousand line program in a matter of minutes can be viewed as somewhat awesome and causes one to wonder just where this capability of 'software analyzing software' will ultimately lead.
SUBROUTINE DEFINEFILIES(SOURCEFILE)

FUNCTION NAMES AND CALLS

REQUEST NAME OF FURTHAN SOURCE FILE TO BE ANALYZED

SOURCEFILE =

2 DISPLAY

DISPLAY"ENTER NAME OF FURTHER SOURCE FILE TO BE ANALYZED"

ACCEPT SOURCEFILE

SOURCEFILE =

DISPLAY"INVALID NAME, PLEASE RENTER."

GO TO 1

DEFINE PERMANENT FILE FOR REQUESTED SOURCE FILE NAME (TO BE
DELETED AFTER SOURCE CODE IS REARRANGED)

FORMAT("FILE FMT:64","AM",".UD",.1DC)

CALL COMMAND(ASC,REH,NPARK)

DEFINE PERMANENT FILE FOR REQUESTED SOURCE FILE NAME (TO STORE
REARRANGED CODE AND TO BE USED AT COMPLETION OF PROGRAM)

hitlazed,9974:SOURCEFILE

SOURCEFILE("FILE FMT:64",AM,\"HITLIAZED-90,\",ASC\",\",DIRE\",\",DIRE\",\",DIRE\",\",DIRE\",\",DIRE\",\",DIRE\",\",END\")

1,1DC)

CALL COMMAND(ASC,REH,NPARK)

MOVE RECORD POINTER TO BEGINNING OF FILE FOR FILE # 40

RENAME 40

WRITE KOF AT BEGINNING OF FILE FOR FILE # 41

CALL UNITCONTROL(41,11)

RETURN

SUBROUTINE MAP

NAME TYPE STRUCTURE ADDRESS
COMMAND SUBROUTINE 0131,1
ASC CHARACTER SIMPLE VAR 0141
NPARK INTEGER SIMPLE VAR 0141
UNITCONTROL SUBROUTINE

LABEL MAP

STATEMENT CODE STATEMENT CODE STATEMENT CODE STATEMENT CODE STATEMENT CODE STATEMENT CODE STATEMENT CODE STATEMENT CODE

1 2 3 4 5 6 7 8 9 10

PROGRAM UNIT DEFINEFILIES COMPILED
C BEGIN TO READ PROGRAM UNIT

C SEGMENT X

COMMUTCOUNTER

FORMATCOUNTER

DECLARECOUNTER

LASTSTATETYPE=

2 LCMLC1

READ44HLC1, LEM69YSTRING

C SKIP SHUNTING LINES IF CURRENT STATE NEXT ON CONTROL STATE

FUNCTION

IF (STRING11111, E2, "C"1 JUMPCOUNTER=1 CURRENTSTATE)

IF (STRING11111, E2, "C") JUMPTO 12

IF (STRING11111, E2, "C") JUMPTO 2

C DETERMINE IF PROGRAM UNIT IS MAIN, FUNCTION, OR SUBROUTINE AND

C STORE BEGINNING LINE # AND PROGRAM UNIT NAME, IF REQUEDED ARM

IF (INDECSTRING, SUN), CT, O, UN, INDEXSTRING, FUNCTION), CT, O, ICALL

C IF SUBROUTINE, STORE SUBROUTINE NAME IN 'SUBNAME' ARM

IF (INDECSTRING, SUN), CT, O, ICALL, SUBNAME(SUBNAMECOUNTER, SUBNAME)

C IF FUNCTION, STORE FUNCTION NAME IN 'FUNCTIONNAME' ARM

IF (INDECSTRING, FUNCTION), CT, O, ICALL, FUNCTIONNAME(FUNCTIONNAME, )

C IF UNSHUNTED, STORE UNSHUNTED NAME IN 'SUBNAME' ARM

C END.
00826 00134000  STRINGIFYUPLAN (STRING, CALL, I, SUB, FUNCTION)  
00826 00150000  IF (INDEX (STRING, CALL, I, SUB, FUNCTION) = 10) 
00826 00162000  IF (INDEX (STRING, CALL, I, SUB, FUNCTION) = 11)  
00826 00184000  IF (INDEX (STRING, CALL, I, SUB, FUNCTION) = 12)  
00826 00196000  IF (INDEX (STRING, CALL, I, SUB, FUNCTION) = 13)  
00826 00178000  IF (INDEX (STRING, CALL, I, SUB, FUNCTION) = 14) 
00826 00200000  IF (INDEX (STRING, CALL, I, SUB, FUNCTION) = 15)  
00826 00232000  1 GO TO 17  
00826 00240000  IF (INDEX (STRING, CALL, I, SUB, FUNCTION) = 16) 
00826 00258000  1 GO TO 11 
00826 00266000  10 CONTINUE  
00826 00274000  GO TO 2  
00826 00282000  11 GO TO 9, 12  
00826 00290000  STRING[I] = " "  
00826 00298000  20 CONTINUE  
00826 00306000  STRING[I] = " " 
00826 00314000  C REMOVE ANY EXTRA BLANKS BEFORE NAME  
00826 00326000  C STRING[I] = " " 
00826 00340000  C IF (INDEX) (STRING, CALL, I, SUB, FUNCTION) = 0) 
00826 00358000  RETURN  
00826 00380000  END  

SYMBOL MAP

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>STRUCTURE</th>
<th>ADDRESS</th>
<th>NAME</th>
<th>TYPE</th>
<th>STRUCTURE</th>
<th>ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CALL</td>
<td>CHARACTER</td>
<td>SIMPLE VAN</td>
<td>U-84 , 1</td>
<td>FUNCTION</td>
<td>CHARACTER</td>
<td>SIMPLE VAN</td>
<td>U-84 , 1</td>
</tr>
<tr>
<td>I</td>
<td>INTEGER</td>
<td>SIMPLE VAN</td>
<td>U+1</td>
<td>INDEX</td>
<td>INTEGER</td>
<td>SIMPLE VAN</td>
<td>U+1</td>
</tr>
<tr>
<td>J</td>
<td>INTEGER</td>
<td>SIMPLE VAN</td>
<td>U-12</td>
<td>RSTRING</td>
<td>CHARACTER</td>
<td>SIMPLE VAN</td>
<td>U+9</td>
</tr>
<tr>
<td>STRING</td>
<td>CHARACTER</td>
<td>SIMPLE VAN</td>
<td>U-61</td>
<td>STRINGIFYPLAN</td>
<td>CHARACTER</td>
<td>SIMPLE VAN</td>
<td>U+9</td>
</tr>
<tr>
<td>SUB</td>
<td>CHARACTER</td>
<td>SIMPLE VAN</td>
<td>U-61</td>
<td>SUBSTRING</td>
<td>CHARACTER</td>
<td>SIMPLE VAN</td>
<td>U+9</td>
</tr>
</tbody>
</table>

LABEL MAP

<table>
<thead>
<tr>
<th>STATEMENT LABEL</th>
<th>CODE OFFSET</th>
<th>STATEMENT LABEL</th>
<th>CODE OFFSET</th>
<th>STATEMENT LABEL</th>
<th>CODE OFFSET</th>
<th>STATEMENT LABEL</th>
<th>CODE OFFSET</th>
<th>STATEMENT LABEL</th>
<th>CODE OFFSET</th>
<th>STATEMENT LABEL</th>
<th>CODE OFFSET</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>103</td>
<td>2</td>
<td>154</td>
<td>1</td>
<td>241</td>
<td>10</td>
<td>114</td>
<td>11</td>
<td>113</td>
<td>40</td>
<td>124</td>
</tr>
</tbody>
</table>

PROGRAM UNIT STRINGIFYPLAN COMPILED
### SYMBOL MAP

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>STRUCTURE</th>
<th>ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DETERMINE_NAME</td>
<td>SUBROUTINE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LEN</td>
<td>INTEGER</td>
<td>SIMPLE</td>
<td>U-4</td>
</tr>
<tr>
<td>NAMECOUNTER</td>
<td>INTEGER</td>
<td>SIMPLE</td>
<td>U-110</td>
</tr>
<tr>
<td>STRING</td>
<td>CHARACTER</td>
<td>SIMPLE</td>
<td>U-45</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>STRUCTURE</th>
<th>ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>INDEXNUM</td>
<td>INTEGER</td>
<td>SIMPLE</td>
<td>U-46</td>
</tr>
<tr>
<td>NAME</td>
<td>CHARACTER</td>
<td>ARRAY</td>
<td>U-47</td>
</tr>
<tr>
<td>STOKENAME</td>
<td>CHARACTER</td>
<td>SIMPLE</td>
<td>U-48</td>
</tr>
<tr>
<td>SUBROUTINE</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

PROGRAM UNIT STOKENAME COMPILED
```plaintext
00010 0017000     SUBROUTINE INIT,MAP,NAME,LOCATION
00010 0017400     SUBROUTINE DELETENAMESLIST,NAME,STRING
00010 0017460     CHARACTER NAME,STRING
00010 0017580     LOAD NAME INTO 'NAME' VARIABLE
00010 0017660     NAME=STRING,LIST,TITLE
00017 0017700     RETURN
00020 0017780     END

SIMUL MAP

NAME      TYPE     STRUCTURE    ADDRESS
DETERMINE/NAME  SUBROUTINE    SIMPLE VAR 0-95 ,1
NAME      CHARACTEN

PROGRAM UNIT DETERMINE/NAME COMPILED
```
00051  00770000  $CUTREX. INIT.MAP, LHL, LGIACUER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
00051  00000000  CUSMOUNT, CLASSTAPL, CURRENTCOUNTER, FORMATCOUNTER.
PAGE 0110  CLASSIFYSTATEMENT

SYMBOL MAP

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>STRUCTURE</th>
<th>ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AASUBSTITUTION</td>
<td>CHARACTEN</td>
<td>ANIAT</td>
<td>0-84</td>
</tr>
<tr>
<td>CALL</td>
<td>CHARACTEN</td>
<td>SIMPLE VAN</td>
<td>0-811</td>
</tr>
<tr>
<td>CLASSIFYSTATEMENT</td>
<td>CHARACTEN</td>
<td>SIMPLE VAN</td>
<td>0-813</td>
</tr>
<tr>
<td>DECLARATION</td>
<td>INTEGR</td>
<td>SIMPLE VAN</td>
<td>0-815</td>
</tr>
<tr>
<td>DETERMINENT</td>
<td>CHARACTEN</td>
<td>SIMPLE VAN</td>
<td>0-843</td>
</tr>
<tr>
<td>EXIT</td>
<td>CHARACTEN</td>
<td>SIMPLE VAN</td>
<td>0-85</td>
</tr>
<tr>
<td>FORMATTLABEL</td>
<td>INTEGR</td>
<td>SIMPLE VAN</td>
<td>0-844</td>
</tr>
<tr>
<td>LC</td>
<td>INTEGR</td>
<td>SIMPLE VAN</td>
<td>0-812</td>
</tr>
<tr>
<td>MARK</td>
<td>INTEGR</td>
<td>SIMPLE VAN</td>
<td>0-846</td>
</tr>
<tr>
<td>MULT</td>
<td>INTEGR</td>
<td>SIMPLE VAN</td>
<td>0-849</td>
</tr>
<tr>
<td>REDUCED</td>
<td>CHARACTEN</td>
<td>SIMPLE VAN</td>
<td>0-85</td>
</tr>
<tr>
<td>SCOPEDIM</td>
<td>INTEGR</td>
<td>SIMPLE VAN</td>
<td>0-810</td>
</tr>
<tr>
<td>STRING</td>
<td>CHARACTEN</td>
<td>SIMPLE VAN</td>
<td>0-810</td>
</tr>
</tbody>
</table>

LABEL MAP

<table>
<thead>
<tr>
<th>STATEMENT LABEL</th>
<th>CODE OFFSET</th>
<th>STATEMENT LABEL</th>
<th>CODE OFFSET</th>
<th>STATEMENT LABEL</th>
<th>CODE OFFSET</th>
<th>STATEMENT LABEL</th>
<th>CODE OFFSET</th>
<th>STATEMENT LABEL</th>
<th>CODE OFFSET</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 FMT 0</td>
<td>5</td>
<td>100</td>
<td>31</td>
<td>220</td>
<td>41</td>
<td>404</td>
<td>614</td>
<td>99</td>
<td>620</td>
</tr>
</tbody>
</table>

PROGRAM UNIT CLASSIFYSTATEMENT COMPILED
0010 00250000  ECONTROL INIT, MAP, LABEL, LOCATION
0010 00230000  SUBROUTINE DETECDISUNIT(FORMAT, FORMATCOUNT, LASTSTATETYPE, CALL, 1, 1, 1)
0010 00210000  CHARACTER LASTSTATETYPE4, CALLS5, STRING12
0010 00220000  INTRIN FFORMATCOUNT
0010 00240000  LASTSTATETYPE=""
0010 00241000  IF(INDEXSTRING("DISPLAY"), GT.0, AND, INDEXSTRING, CALL).EQ.0)
0011 00242000  RETURN 2
0012 00243000  IF(INDEXSTRING("FORMAT"), GT.0, AND, INDEXSTRING, "" FORMAT).EQ.0
0013 00244000  IF(INDEXSTRING("FORMAT"), GT.0, AND, INDEXSTRING, "" FORMAT).EQ.0
0014 00245000  IF(INDEXSTRING("FORMAT"), GT.0, AND, INDEXSTRING, "" FORMAT).EQ.0
0015 00246000  IF(INDEXSTRING("FORMAT"), GT.0, AND, INDEXSTRING, "" FORMAT).EQ.0
0016 00247000  RETURN 1
0017 00248000  RETURN
0018 00249000  RETURN
0019 00250000  END

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>STRUCTURE</th>
<th>ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CALL</td>
<td>CHAR</td>
<td>SIMPLE</td>
<td>SND</td>
</tr>
<tr>
<td>FORMATCOUNT</td>
<td>INT</td>
<td>SIMPLE</td>
<td>SNL</td>
</tr>
<tr>
<td>STRING</td>
<td>CHAR</td>
<td>SIMPLE</td>
<td>SND</td>
</tr>
</tbody>
</table>

PROGRAM UNIT DETECDISUNIT COMPILED
**Statement Map**

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>STRUCTURE</th>
<th>ADDRESS</th>
<th>NAME</th>
<th>TYPE</th>
<th>STRUCTURE</th>
<th>ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CALL</td>
<td>CHARACTER</td>
<td>SIMPLE VAN</td>
<td>U-110</td>
<td>FUNCTION</td>
<td>CHARACTER</td>
<td>SIMPLE VAN</td>
<td>U-34</td>
</tr>
<tr>
<td>I</td>
<td>INTEGER</td>
<td>SIMPLE VAN</td>
<td>U+11</td>
<td>ISTART</td>
<td>INTEGER</td>
<td>SIMPLE VAN</td>
<td>U-32</td>
</tr>
<tr>
<td>LC</td>
<td>INTEGER</td>
<td>SIMPLE VAN</td>
<td>U-55</td>
<td>REMSTRING</td>
<td>CHARACTER</td>
<td>SIMPLE VAN</td>
<td>U-64</td>
</tr>
<tr>
<td>NEXTSTRING</td>
<td>INTEGER</td>
<td>SIMPLE VAN</td>
<td>U-45</td>
<td>PROCESSIF</td>
<td>CHARACTER</td>
<td>SIMPLE VAN</td>
<td>U-31</td>
</tr>
<tr>
<td>PROCESSCALL</td>
<td>CHARACTER</td>
<td>ARRAY</td>
<td>U-18</td>
<td>SUBSTRING</td>
<td>CHARACTER</td>
<td>SIMPLE VAN</td>
<td>U-64</td>
</tr>
<tr>
<td>REDUCED</td>
<td>CHARACTER</td>
<td>ARRAY</td>
<td>U-15</td>
<td>SUBSTRING</td>
<td>CHARACTER</td>
<td>SIMPLE VAN</td>
<td>U-64</td>
</tr>
<tr>
<td>STRIPUPPARAM</td>
<td>SUBROUTING</td>
<td></td>
<td></td>
<td></td>
<td>SUBROUTING</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Label Map**

<table>
<thead>
<tr>
<th>STATEMENT</th>
<th>CODE</th>
<th>STATEMENT</th>
<th>CODE</th>
<th>STATEMENT</th>
<th>CODE</th>
<th>STATEMENT</th>
<th>CODE</th>
<th>STATEMENT</th>
<th>CODE</th>
<th>STATEMENT</th>
<th>CODE</th>
<th>STATEMENT</th>
<th>CODE</th>
<th>STATEMENT</th>
<th>CODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
<td>10</td>
<td>11</td>
<td>225</td>
<td>12</td>
<td>245</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*PROGRAM UNIT PROCESSCALL COMPILED*
00058  00274000  BEGIN  IF  IFF=1, STRING111, ELSE STRING114, CALL"
00061  00279000  IF IFF=1
00064  00285000  IF IFF=1 STRING111, ELSE STRING114, CALL"
00067  00289000  DISPLAY"EXPECTED IF STATEMENT WITH CALL, INSTEAD GET"
00071  00293000  DISPLAY STRING111
00075  00297000  DISPLAY STRING114
00079  00299000  DISPLAY"THUS PROGRAM IS TERMINATED."
00082  00301000  STUP
00085  00303000  C REMOVE ARGUMENT UF IF STATEMENT
00088  00305000  I ISTART=INDEXSTRING, CALL"
00091  00309000  STRING111=IFF
00093  00309000  STRING111=" "
00095  00310000  STRING111=STRING114, ISTART=STRING114"
00098  00311000  RETURN
00101  00312000  END

SYMBOL MAP

NAME   TYPE   STRUCTURE   ADDRESS
CALL   CHARACTER   SIMPLE   VAN   U-64   4
ISTART  INTEGER   SIMPLE   VAN   U-61   4
PROCESSIV   SUBROUTINE

LABEL MAP

STATEMENT   CODE   STATEMENT   CODE   STATEMENT   CODE   STATEMENT   CODE   STATEMENT   CODE
LABEL   OFFSET   LABEL   OFFSET   LABEL   OFFSET   LABEL   OFFSET   LABEL   OFFSET
1        100

PROGRAM UNIT PROCESSIV COMPILED.
C subroutine determines if statement type is declarative

CHARACTER LASTSTATEMENT, STRING77, STRING

COMMON COUNT1, ULCOUNT1, EXTC

LASTSTATEMENT = "ul"

OLDCOUNT1 = COUNT1

IF (INDEXSTRING("PROGRAM ", 1, COUNT1)) THEN
  COUNT1 = ULCOUNT1
ENDIF

IF (INDEXSTRING("FUNCTION ", 1, COUNT1)) THEN
  COUNT1 = ULCOUNT1
ENDIF

IF (INDEXSTRING("REAL ", 1, COUNT1)) THEN
  COUNT1 = ULCOUNT1
ENDIF

IF (INDEXSTRING("INTEGER ", 1, COUNT1)) THEN
  COUNT1 = ULCOUNT1
ENDIF

IF (INDEXSTRING("LOGICAL ", 1, COUNT1)) THEN
  COUNT1 = ULCOUNT1
ENDIF

IF (INDEXSTRING("COMPLEX ", 1, COUNT1)) THEN
  COUNT1 = ULCOUNT1
ENDIF

RETURN

END
0001 00341000 DECLARE INIT, MAP, LABEL, LOCATION
0001 00349200 SUBROUTINE DECLRTIELCALLS(GTNAME, SUINAME, CURRENT, CALL, SUB, REDUCED, 
0001 00349600 INLC(FUNCTION) 
0001 00349800 CHARACTER$ GTNAME$'140', STRING$'2', CALL$'5', NAME$'6', SUB$'12', 
0001 00349C00 INCLUD$'21500', FUNCTION$'8' 
0001 0034A000 INCLUDE DECLRTIELCALLS, CURRENT, LOCATION, MAP, CALL, CLASS 
0001 0034A400 DECLARE CURRENT, GTNAME, CALL 
0001 0034A600 LC40 
0001 0034A800 GO 20 J=1, MLC 
0001 0034AA00 READ(REDCIDU(J), YSTRING, SEGBEGIN, SEGEND, CURRENT, LOCATION, 
0001 0034AC00 INLC(FUNCTION), DECLARE CURRENT, MLC 
0011 0034D200 Y FORMAT(AF2, 318) 
0011 0034E00 C DETERMINE IF PROGRAM UNIT STATEMENT 
0011 0034E400 IF(STRING(114), GT$'100', CALL$'510') THEN DECLARE(MLC(J), LOCATION, STRING, SEGBEGIN, 
0011 0034E800 DECLARE, CURRENT, LOCATION, MLC, DECLARE CURRENT, MLC 
0011 0034EC00 IF(STRING, GT$'195', GT$'95', CALL$'510') THEN STRING, SEGBEGIN, 
0011 0034F00 DECLARE, CURRENT, LOCATION, MLC, DECLARE CURRENT, MLC 
0024 00350000 IF(STRING, FUNCTION$'8', GT$'0', STRING$'1', CALL$'510') THEN STRING, 
0024 00350400 DECLARE, CURRENT, LOCATION, MLC, DECLARE CURRENT, MLC 
0033 00350800 IF(STRING, GT$'100', GT$'95', GT$'1100', STRING$'1', CALL$'510') THEN STRING, 
0033 00350C00 DECLARE, CURRENT, LOCATION, MLC, DECLARE CURRENT, MLC 
0041 00351000 IF(STRING, GT$'100', GT$'95', STRING$'1', CALL$'510') THEN STRING, 
0041 00351400 DECLARE, CURRENT, LOCATION, MLC, DECLARE CURRENT, MLC 
0047 00351800 C DETERMINED TO BE 'CALL' STATEMENT; DETERMINE BEGINNING OF CALL NAME 
0047 00351C00 INCLUD$'21500', STRING, SEGBEGIN 
0047 00352000 Declare calls, MLC(J) 
0047 00352400 C DETERMINE NAME OF CALL 
0047 00352800 CALL DETERMINE NAME, GTNAME, NAME, STRING(J) 
0047 00352C00 C DETERMINE IF CALL NAME EXISTS AS A SUBROUTINE NAME 
0047 00353000 GO 10 U, SUINAMECOUNT 
0047 00353400 IF(NAME, EQ, SUINAME, 'GTNAME,J', STRING, SEGBEGIN, SEGEND 
0051 00353800 IF(NAME, EQ, SUINAME, 'GTNAME, L', SEGEND 
0055 00353C00 IF(NAME, EQ, SUINAME, 'GTNAME, J', SEGEND 
0057 00354000 1 CONTINUE 
0059 00354400 C SUBROUTINE NAME NOT FOUND; THEREFORE, CALL STATEMENT NOT WRITTEN TO 
0059 00354800 L CONTINUE 
005C 00355000 C 'RENUCTD' ARRAY 
005C 00355400 @2C000000 
005C 00355800 20 CONTINUE 
005C 00355C00 99 MLC(J) 
005C 00356000 RETURN 
005C 00356400 END 

SYMBOL MAP

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>STRUCTURE</th>
<th>ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CALL</td>
<td>CHARACTER</td>
<td>SIMPLE VAR</td>
<td>0-112, 1</td>
</tr>
<tr>
<td>DECLARECURRENT</td>
<td>INTEGER</td>
<td>SIMPLE VAR</td>
<td>0120</td>
</tr>
<tr>
<td>DETERMINENAME</td>
<td>INTEGER</td>
<td>SIMPLE VAR</td>
<td>0124</td>
</tr>
<tr>
<td>FUNCTION</td>
<td>CHARACTER</td>
<td>SIMPLE VAR</td>
<td>0-112, 1</td>
</tr>
<tr>
<td>INITENUM</td>
<td>INTEGER</td>
<td>SIMPLE VAR</td>
<td>0-112, 1</td>
</tr>
<tr>
<td>LC</td>
<td>INTEGER</td>
<td>SIMPLE VAR</td>
<td>0-112, 1</td>
</tr>
<tr>
<td>NAME</td>
<td>CHARACTER</td>
<td>SIMPLE VAR</td>
<td>0-112, 1</td>
</tr>
<tr>
<td>subclasses</td>
<td>INTEGER</td>
<td>SIMPLE VAR</td>
<td>0-112, 1</td>
</tr>
<tr>
<td>STRING</td>
<td>CHARACTER</td>
<td>ARRAY</td>
<td>0-112, 1</td>
</tr>
</tbody>
</table>
### SYMBOL MAP

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>STRUCTURE</th>
<th>ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CALL</td>
<td>CHARACTER</td>
<td>SIMPLE VAR</td>
<td>$0-95$</td>
</tr>
<tr>
<td>DECLARATIONS</td>
<td>CHARACTER</td>
<td>SIMPLE VAR</td>
<td>$0-46$</td>
</tr>
<tr>
<td>FUNCTIONCOUNTER</td>
<td>INTEGER</td>
<td>SIMPLE VAR</td>
<td>$0-11$</td>
</tr>
<tr>
<td>INSERTFUNTCALLS</td>
<td>CHARACTER</td>
<td>SIMPLE VAR</td>
<td>$0-49$</td>
</tr>
<tr>
<td>LASTSTATE</td>
<td>CHARACTER</td>
<td>SIMPLE VAR</td>
<td>$0-11$</td>
</tr>
<tr>
<td>REduced</td>
<td>CHARACTER</td>
<td>ARRAY</td>
<td>$0-11$</td>
</tr>
<tr>
<td>STRING</td>
<td>CHARACTER</td>
<td>SIMPLE VAR</td>
<td>$0-112$</td>
</tr>
</tbody>
</table>

### LABEL MAP

<table>
<thead>
<tr>
<th>STATEMENT</th>
<th>CODE OFFSET</th>
<th>STATEMENT</th>
<th>CODE OFFSET</th>
<th>STATEMENT</th>
<th>CODE OFFSET</th>
<th>STATEMENT</th>
<th>CODE OFFSET</th>
<th>STATEMENT</th>
<th>CODE OFFSET</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 FMT</td>
<td>0</td>
<td>2</td>
<td>121</td>
<td>3</td>
<td>241</td>
<td>10</td>
<td>5/2</td>
<td>11</td>
<td>46/24</td>
</tr>
<tr>
<td>29</td>
<td>111</td>
<td>30</td>
<td>122</td>
<td>31</td>
<td>123</td>
<td>17 FMT</td>
<td>4</td>
<td>99</td>
<td>5/3</td>
</tr>
</tbody>
</table>

PROGRAM UNIT INSERTFUNTCALLS COMPILED
DATA

NAME      TYPE  STRUCTURE   ADDRESS  NAME      TYPE  STRUCTURE   ADDRESS
KEYLENGTH  INTEGER  SIMPLE VAR  U-41   JJ     INTEGER  SIMPLE VAR  U-52
NLC       INTEGER  SIMPLE VAR  U-49   JJ     INTEGER  SIMPLE VAR  U-55
SORTARRAY  INTEGER  SIMPLE VAR  U-46   JJ     INTEGER  SIMPLE VAR  U-47

LABEL MAP

STATEMENT CODE STATEMENT CODE STATEMENT CODE STATEMENT CODE STATEMENT CODE STATEMENT CODE STATEMENT CODE STATEMENT CODE
LABEL OFFSET LABEL OFFSET LABEL OFFSET LABEL OFFSET LABEL OFFSET LABEL OFFSET LABEL OFFSET LABEL OFFSET LABEL OFFSET
100     J0  101     J1

PROGRAM UNIT SORTARRAY COMPILED
```
<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>STRUCTURE</th>
<th>ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECORDJ</td>
<td>CHARACTER</td>
<td>SIMPLE VAN U-85</td>
<td>1</td>
</tr>
<tr>
<td>Switch</td>
<td>CHARACTER</td>
<td>SUBROUTINE</td>
<td></td>
</tr>
</tbody>
</table>

PROGRAM UNIT SWITCH COMPILED
```
00120 00511000 $CONTROL INIT, MAP, LABLE, LOCATION
00120 00511400 SUBROUTINCE: CLEVERSE, SUFFIX, CALL, ARGSUBSTITUTINS, LLC
00120 00513000 CHARACTER STRING, CALLS, NAME, SUFFIX, REDUCED, REDUCED, CALLS
00120 00516000 IF FUNCTION=10, ARGSUBSTITUTIONS=10, 0, 8, ARGNAME=16, SURNAME=8
00120 00518000 INTEGER SEGLEN, SEGLEN, LLC
00120 00519000 LEDC
00122 00520000 QLEN=NC
00124 00521000 DD 20=2, LLC
00124 00522000 READ(REDUCED, Y, STRING, SEGLEN, LLC
00124 00523000 Y FUNCTION=12, 24)
00124 00524000 C DETERMINE IF CALL STATEMENT IS WITHIN PROGRAM UNIT WITH EXTERNAL STATEMENT
00124 00525000 IF (STRING, CALL, CT. S AND, LLC, CT. TO) TO 11
00124 00526000 DD TO 20
00124 00527000 II ISTRAPungle(STRING, CALL)
00212 00528000 NAME=STRING(ISTRAPungle)
00221 00529000 C FIND LOCATION OF SUBROUTINE BY SERIALLY READING "REDUCED" FILE
00221 00530000 WRITE(STRING(ISTRAPungle), LLC, NAME
00225 00531000 DD 16=16, LLC
00225 00532000 DD 16=16, LLC
00225 00533000 READ(REDUCED(II), Y, STRING, SEGLEN, LLC
00310 00534000 IF (STRING(ISTRAPungle), CT. ZERO) TO 12
00322 00535000 10 CONTINUE
00322 00536000 DISPLAY NAME, NOT FOUND AS SUBROUTINE IN "REDUCED" FILE, ERROR
00400 00537000 STOP
00400 00538000 C READ NAME OF CALL STATEMENT AND ITS LIST POSITION FROM
00400 00539000 C ARGSUBSTITUTION ARRY
00400 00540000 12 DD 100=2, 1, 1
00400 00541000 READ(ARGSUBSTITUTION(LRC, 0), 1, 1, LLC, ARGNAME, LISTPOS
00444 00543000 11 DD (ISTRAPungle, LLC, 0, 1, 12)
00446 00544000 11 DD (ISTRAPungle, LLC, 0, 1, 12)
00451 00545000 C GET ARGUMENT NAME FROM SAME LIST POSITION OF SUBROUTINE STATEMENT IN
00451 00546000 C SOURCE FILE AND STORE IN "ARGSUBSTITUTION" ARRY AND FIND WHERE
00453 00547000 C CALLED IN THIS CALLD SUBROUTINE
00453 00547000 CALL GETSUBNAME(ARGSUBSTITUTION, ARGNAME, LISTPOS, SEGLEN, LLC
00453 00548000 LLCCALL.CALL, CALL, CALL, SUFFIX, NAME, LLC, SURNAME=8
00471 00549000 C ADD SUBNAME TO "REDUCED" FILE IF IT IS NOT ALREADY THERE
00471 00550000 IF (LRC, LLC, 0), 1 TO 987
00475 00551000 DD 900=9000, LLC
00503 00552000 READ(REDUCED(II), Y, STRING, SEGLEN, LLC
00534 00553000 IF (STRING(ISTRAPungle), CT. ZERO) TO 100
00551 00554000 999 CONTINUE
00552 00555000 987=LENGTH+1, LLC
00553 00556000 %WRITE(REDUCED, LLC, 0, )SUBNAME, LLLCALL
00604 00557000 VI FUNCTION="CALL", A10, "PASSED VIA EXTERNAL", 00, 14)
00604 00558000 19 CONTINUE
00605 00559000 20 CONTINUE
00612 00560000 C SORT REDUCED FILE SO THAT JUST INSULTED CALLS ARE IN SAME SUBNAME
00612 00561000 C IN REDUCED ARRAYS AS THEY ARE IN SOURCE CODE FILE
00612 00560000 CALL SORTNAME(REDUCED, LLC, Z, 4)
00624 00562000 RETURN
00625 00564000 END
## SYMBOL MAP

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>STRUCTURE</th>
<th>ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARGNAME</td>
<td>CHARACTER</td>
<td>SIMPLE</td>
<td>VAN U+915 .1</td>
</tr>
<tr>
<td>CALL</td>
<td>CHARACTER</td>
<td>SIMPLE</td>
<td>VAN U+916 .1</td>
</tr>
<tr>
<td>CONSNAME</td>
<td>CHARACTER</td>
<td>SIMPLE</td>
<td>VAN U+921 .1</td>
</tr>
<tr>
<td>GETSUBNAME</td>
<td>SUBROUTINE</td>
<td>SIMPLE</td>
<td>U+922 .1</td>
</tr>
<tr>
<td>LSTART</td>
<td>INTEGER</td>
<td>SIMPLE</td>
<td>VAN U+923</td>
</tr>
<tr>
<td>LISTPUS</td>
<td>INTEGER</td>
<td>SIMPLE</td>
<td>VAN U+924</td>
</tr>
<tr>
<td>LOCCELL</td>
<td>INTEGER</td>
<td>SIMPLE</td>
<td>VAN U+925</td>
</tr>
<tr>
<td>NAME</td>
<td>CHARACTER</td>
<td>SIMPLE</td>
<td>VAN U+926 .1</td>
</tr>
<tr>
<td>ULC</td>
<td>INTEGER</td>
<td>SIMPLE</td>
<td>VAN U+927 .1</td>
</tr>
<tr>
<td>SGNBEGIN</td>
<td>INTEGER</td>
<td>SIMPLE</td>
<td>VAN U+928</td>
</tr>
<tr>
<td>SORTARRAT</td>
<td>SUBROUTINE</td>
<td>SIMPLE</td>
<td>U+929 .1</td>
</tr>
<tr>
<td>SUB</td>
<td>CHARACTER</td>
<td>SIMPLE</td>
<td>VAN U+930 .1</td>
</tr>
</tbody>
</table>

## LABEL MAP

<table>
<thead>
<tr>
<th>STATEMENT LABEL</th>
<th>CODE OFFSET</th>
<th>STATEMENT LABEL</th>
<th>CODE OFFSET</th>
<th>STATEMENT LABEL</th>
<th>CODE OFFSET</th>
<th>STATEMENT LABEL</th>
<th>CODE OFFSET</th>
<th>STATEMENT LABEL</th>
<th>CODE OFFSET</th>
<th>STATEMENT LABEL</th>
<th>CODE OFFSET</th>
<th>STATEMENT LABEL</th>
<th>CODE OFFSET</th>
</tr>
</thead>
<tbody>
<tr>
<td>9 PAT</td>
<td>0</td>
<td>10 PAT</td>
<td>22</td>
<td>11</td>
<td>24</td>
<td>12</td>
<td>401</td>
<td>20</td>
<td>562</td>
<td>45 PAT</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7 PAT</td>
<td>12</td>
<td>9 PAT</td>
<td>21</td>
<td>100</td>
<td>404</td>
<td>750</td>
<td>551</td>
<td>967</td>
<td>552</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

PROGRAM UNIT COMPLETEANGLIST COMPILED
00044 00560000 53CABINE INIT.ARRL.NE4.LOCATION
00044 00560000 C THIS SUBROUTINE FINDS ARGUMENT NAME IN SUBROUTINE STATEMENT
00044 00560000 C THAT THE CALLING PROGRAM UNIT HAS DECLARED AS "EXTERNAL"
00044 00560000 C AND STORES IT IN "ARGSUBSTITUTION" ARRAY.
00044 00560000 C THEN THE CALL STATEMENT USING THIS ARGUMENT IS FOUND AND ITS LOCATION
00044 00560000 C IN THE SOURCE CODE FILE RELOCATED.
00044 00570000 SUBROUTINE CTOARGNAME,ARGSUBSTITUTION,ARGNAME,LISTNAME,AUXNAME.
00044 00572000 ILOCAL,CALLSUB,ARGNAME,LISTNAME,AUXNAME.
00044 00573000 C CCHARACTER STRING1(),ARGSUBSTITUTION,500(10,4),SUBNAME(10).
00044 00574000 ICALLSUB,ARGNAME,ARGNAME,CALLSTRING1(),ARGSUBSTITUTION,1000,LISTNAME,10.
00044 00575000 INTEGER FOUNDO,ARGNAME.
00044 00576000 LC=ARGNAME=1
00047 00577000 99 LC=LC+1
00050 00578000 C FIND SUBROUTINE STATEMENT
00050 00578000 HEAD(LOC1,111)STRING
00074 00580000 111 FORMAT(A12)
00074 00581000 IF (LENNAME(STRING),1406)10 FOUNDO 99
00105 00582000 C GET ARGUMENT NAME FROM ARGUMENT LIST OF SUBROUTINE STATEMENT
00105 00583000 FOUNDO
00107 00584000 DO 400 K=1,10,string(STRING(),1)
00121 00585000 IF (STRENGLK(),MK+1,MK+1)10,6,400
00161 00586000 FOUT=FOUNDO+1
00162 00587000 IF (FOUND=EO,TOPOFMU)10,99,99
00167 00588000 400 CONTINUE
00170 00589000 IF (KNAME(K))10,21,21
00172 00590000 DO 400 K=1,10,NAME(K)
00201 00591000 IF (NAME(),MK+1,MK+1)10,6,400
00242 00592000 40 CONTINUE
00241 00593000 31 NNAME=NAME=1
00246 00594000 SUBNAME=STRING1(1),NNAME(K)
00255 00595000 SUBNAME(SUBNAME(K),1)=SUBNAME(K)
00258 00596000 C STRING IMP IN ARGSUBSTITUTION ARRAY.
00270 00597000 WHILE(ARGSUBSTITUTION(LOC1,71),SUBNAME,ARGNAME,SUBNAME,LISTNAME)
00274 00598000 77 FORMAT(A12)
00274 00599000 10 CONTINUE
00277 00600000 10 CONTINUE
00347 00600000 C FIND CALL STATEMENT THAT HAS THIS SUBNAME.
00347 00601000 WRITE(1CNAME(1),1N11,1)
00376 00602000 66 FORMAT(A12)
00376 00603000 DO 400 K=1,1000
00404 00604000 READ(400,110,NAME(STRING)),NAME(STRING).
00433 00605000 IF (INDEX(STRING(STRING,K),1)<>1),GT.400,100
00433 00606000 100
00433 00607000 100 CONTINUE
00433 00608000 404 LISTCALL=1
00456 00609000 96 RETURN
00457 00610000 99 DISPLAY"END OF SOURCE CODE FILE IN SUBROUTINE COMPELAP.C, ERROR"
00511 00611000 STOP
00514 00612000 END

SINONIR MAP

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>STRUCTURE ADDRESS</th>
<th>NAME</th>
<th>TYPE</th>
<th>STRUCTURE ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANCHRAN</td>
<td>CHARACTER</td>
<td>SIMPLE YAN U-111 ,</td>
<td>ANCSRTHSST</td>
<td>CHARACTER</td>
<td>ARRANGEMENT U-619</td>
</tr>
</tbody>
</table>
```plaintext
<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>STRUCTURE</th>
<th>ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISPLAYHPRNAME</td>
<td>SUBUNIT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FUNCTIONCOUNTER</td>
<td>INTEGER</td>
<td>SIMPLE VAR</td>
<td>U-46, 1</td>
</tr>
<tr>
<td>J</td>
<td>INTEGER</td>
<td>SIMPLE VAR</td>
<td>U-42</td>
</tr>
<tr>
<td>NLC</td>
<td>INTEGER</td>
<td>SIMPLE VAR</td>
<td>U-44</td>
</tr>
<tr>
<td>NUNITALUNITS</td>
<td>INTEGER</td>
<td>SIMPLE VAR</td>
<td>U-46</td>
</tr>
<tr>
<td>NAME</td>
<td>TYPE</td>
<td>STRUCTURE</td>
<td>ADDRESS</td>
</tr>
<tr>
<td></td>
<td>SUBUNIT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FUNCTIONNAME</td>
<td>CHARACTER</td>
<td>ARRAY</td>
<td>U-44, 1</td>
</tr>
<tr>
<td>J</td>
<td>INTEGER</td>
<td>SIMPLE VAR</td>
<td>U-44</td>
</tr>
<tr>
<td>NAMELIST</td>
<td>CHARACTER</td>
<td>ARRAY</td>
<td>U-45</td>
</tr>
<tr>
<td>NAME</td>
<td>TYPE</td>
<td>STRUCTURE</td>
<td>ADDRESS</td>
</tr>
<tr>
<td></td>
<td>SUBUNIT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NAMELIST</td>
<td>CHARACTER</td>
<td>ARRAY</td>
<td>U-45</td>
</tr>
</tbody>
</table>
```

```
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
00081 00610000 SUBUNIT NAME=SUBLNAME
```
```
00020  00651000  INCLUDE INIT,MAP,LABEL,LONGNAME
00020  00651000  SUBROUTINE SWISE(MCCURDJ,RECURDJ)
00020  00654000  CHARACTER RECURDJ*16,RECURDJ1*16,TEM*16
00020  00658000  RECURDJ
00020  00657000  RECURDJ1
00020  00659000  RETURN
00020  00659000  END

SYMBOL MAP

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>STRUCTURE</th>
<th>ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECURDJ</td>
<td>CHARACTER</td>
<td>SIMPLE</td>
<td>U-$$</td>
</tr>
<tr>
<td>SWISE</td>
<td>SUBROUTINE</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

PROGRAM UNIT SWISE COMPILED
```
PAGE 0031  HEWLETT-PACKARD 3210A, 01.09  FUTURE/3000  Wed, Jun 15, 1983, 12:10 PM

00031  00660000  EXECUTE UNIT, MAP, LABEL, LOCATION
00035  00661000  SUBROUTINE DISPLAYUPNAMES, DISPLAYUPNAMES, NAME, NAME
00039  00662000  CHARACTER NAME(10), NUMBER(92), STRING(72)
00043  00663000  INTEGER SEGREG, SEGEND
00045  00664000  C GET BEGINNING AND ENDING LINE NUMBERS OF DUPLICATE NAMED PROGRAM
00049  00665000  C UNIT
00053  00666000  DU SU J#, ML
00056  00667000  READ(MLDUCCD(1), 77) STRING, SEGBEGIN, SEGEND
00059  00668000  77 FORMAT(A72,14)
00063  00669000  IF(INDEX(STRING, NAME)) .EQ. 0 THEN Tu 50
00066  00670000  IF(INDEX(STRING, "NAME") .EQ. 0 THEN Tu 50
00069  00671000  DU SEGEND = STRING, "NAME", (INDEX("NAME", "NAME", "SEGEND", "") IS DUPLICATE)
00073  00672000  1, "NAME.
00077  00673000  50 CONTINUE
00081  00674000  RETURN
00085  00675000  END

SIMUL MAP

NAME  TYPE  STRUCTURE  ADDRESS  NAME  TYPE  STRUCTURE  ADDRESS
DISPLAYUPNAMES SUBROUTINE J
NAME  CHARACTER  SIMPLE  VAR  U=15  ,1  NUMBER  CHARACTER  SIMPLE  VAR  U=16  ,1
MLDUCCD  CHARACTER  ARRAY  U=16  ,1  SEGREG  INTEGER  SIMPLE  VAR  U=11  ,1
SEGEND  INTEGER  SIMPLE  VAR  U=11

LABEL MAP

STATEMENT  CODE  STATEMENT  CODE  STATEMENT  CODE  STATEMENT  CODE  STATEMENT  CODE
LABEL  OFFSET  LABEL  OFFSET  LABEL  OFFSET  LABEL  OFFSET  LABEL  OFFSET  LABEL  OFFSET
50  221  77  FAT  0

PROGRAM UNIT DISPLAYUPNAMES COMPILED
### STAGE 1 MAP

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>STRUCTURE</th>
<th>ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANGQSUBSTITION</td>
<td>CHARACTER ARRAY</td>
<td>U-110 ,1</td>
<td></td>
</tr>
<tr>
<td>CALL</td>
<td>CHARACTER SUBROUTINE</td>
<td>U-120 ,1</td>
<td></td>
</tr>
<tr>
<td>DISPLAYCALLS</td>
<td>FUNCTION</td>
<td>U-130 ,1</td>
<td></td>
</tr>
<tr>
<td>FUNCTIONCALL</td>
<td>CHARACTER ARRAY</td>
<td>U-140 ,1</td>
<td></td>
</tr>
<tr>
<td>FUNCTIONNAME</td>
<td>CHARACTER ARRAY</td>
<td>U-150 ,1</td>
<td></td>
</tr>
<tr>
<td>INDEXER</td>
<td>INTEGER SIMPLE VALUE</td>
<td>U-160 ,1</td>
<td></td>
</tr>
<tr>
<td>NUCALL</td>
<td>INTEGER SIMPLE VALUE</td>
<td>U-170 ,1</td>
<td></td>
</tr>
<tr>
<td>PROCUNIT</td>
<td>CHARACTER SIMPLE VALUE</td>
<td>U-180 ,1</td>
<td></td>
</tr>
<tr>
<td>STRING</td>
<td>CHARACTER SIMPLE VALUE</td>
<td>U-190 ,1</td>
<td></td>
</tr>
<tr>
<td>SUBCALL</td>
<td>CHARACTER ARRAY</td>
<td>U-200 ,1</td>
<td></td>
</tr>
<tr>
<td>SUBNAMECOUNT</td>
<td>INTEGER SIMPLE VALUE</td>
<td>U-210 ,1</td>
<td></td>
</tr>
</tbody>
</table>

### TYPE MAP

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>STRUCTURE</th>
<th>ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>A2A</td>
<td>INTEGER SIMPLE VALUE</td>
<td>U-61 ,1</td>
<td></td>
</tr>
<tr>
<td>UNTERMINATEDCALL</td>
<td>FUNCTION</td>
<td>U-62 ,1</td>
<td></td>
</tr>
<tr>
<td>FUNCTIONCALL</td>
<td>INTEGER SIMPLE VALUE</td>
<td>U-63 ,1</td>
<td></td>
</tr>
<tr>
<td>FUNCTIONCOUNTER</td>
<td>INTEGER SIMPLE VALUE</td>
<td>U-64 ,1</td>
<td></td>
</tr>
<tr>
<td>J</td>
<td>INTEGER SIMPLE VALUE</td>
<td>U-65 ,1</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>CHARACTER SIMPLE VALUE</td>
<td>U-66 ,1</td>
<td></td>
</tr>
<tr>
<td>MUCALL</td>
<td>INTEGER SIMPLE VALUE</td>
<td>U-67 ,1</td>
<td></td>
</tr>
<tr>
<td>REDUCED</td>
<td>CHARACTER ARRAY</td>
<td>U-68 ,1</td>
<td></td>
</tr>
<tr>
<td>SUB</td>
<td>CHARACTER SIMPLE VALUE</td>
<td>U-69 ,1</td>
<td></td>
</tr>
<tr>
<td>SUBNAME</td>
<td>CHARACTER ARRAY</td>
<td>U-70 ,1</td>
<td></td>
</tr>
</tbody>
</table>

### LABEL MAP

<table>
<thead>
<tr>
<th>STATEMENT</th>
<th>CODE OFFSET</th>
<th>STATEMENT</th>
<th>CODE OFFSET</th>
<th>STATEMENT</th>
<th>CODE OFFSET</th>
<th>STATEMENT</th>
<th>CODE OFFSET</th>
<th>STATEMENT</th>
<th>CODE OFFSET</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>415</td>
<td>20</td>
<td>241</td>
<td>70</td>
<td>257</td>
<td>11</td>
<td>PMT</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
### Symbol Map

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>STRUCTURE</th>
<th>ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDRESSSTACK</td>
<td>INTEGER</td>
<td>ARRAY</td>
<td>0:61:1</td>
</tr>
<tr>
<td>CALL</td>
<td>CHARACTER</td>
<td>SIMPLE</td>
<td>0:61:1</td>
</tr>
<tr>
<td>FINDPROGRAMUNIT</td>
<td>INTEGER</td>
<td>SUBROUTINE</td>
<td></td>
</tr>
<tr>
<td>İDENTSUBNAME</td>
<td>INTEGER</td>
<td>SUBROUTINE</td>
<td></td>
</tr>
<tr>
<td>LASTLEVEL</td>
<td>INTEGER</td>
<td>ARRAY</td>
<td>0:62:1</td>
</tr>
<tr>
<td>LEXELC</td>
<td>INTEGER</td>
<td>SIMPLE</td>
<td>0:62:1</td>
</tr>
<tr>
<td>NAME</td>
<td>CHARACTER</td>
<td>SIMPLE</td>
<td>0:62:1</td>
</tr>
<tr>
<td>NLC</td>
<td>INTEGER</td>
<td>SIMPLE</td>
<td>0:62:1</td>
</tr>
<tr>
<td>NUMBER DECL</td>
<td>INTEGER</td>
<td>SIMPLE</td>
<td>0:62:1</td>
</tr>
<tr>
<td>NOUNPUTLINES</td>
<td>INTEGER</td>
<td>SIMPLE</td>
<td>0:62:1</td>
</tr>
<tr>
<td>PAGEHEAD</td>
<td>INTEGER</td>
<td>SUBROUTINE</td>
<td></td>
</tr>
<tr>
<td>PRINTLINE</td>
<td>CHARACTER</td>
<td>ARRAY</td>
<td>0:64:1</td>
</tr>
<tr>
<td>RECPULLD</td>
<td>CHARACTER</td>
<td>ARRAY</td>
<td>0:62:1</td>
</tr>
<tr>
<td>SATENFILE</td>
<td>INTEGER</td>
<td>SUBROUTINE</td>
<td></td>
</tr>
<tr>
<td>SEGENDO</td>
<td>INTEGER</td>
<td>SIMPLE</td>
<td>0:62:1</td>
</tr>
<tr>
<td>STARTPHI</td>
<td>INTEGER</td>
<td>SIMPLE</td>
<td>0:62:1</td>
</tr>
<tr>
<td>SUB</td>
<td>INTEGER</td>
<td>SIMPLE</td>
<td>0:62:1</td>
</tr>
<tr>
<td>TITULC</td>
<td>REAL</td>
<td>SIMPLE</td>
<td>0:65:1</td>
</tr>
<tr>
<td>TOTLINES</td>
<td>REAL</td>
<td>SIMPLE</td>
<td>0:66:1</td>
</tr>
<tr>
<td>USED</td>
<td>INTEGER</td>
<td>ARRAY</td>
<td>0:68:1</td>
</tr>
</tbody>
</table>

### Label Map

<table>
<thead>
<tr>
<th>STATEMENT LABEL</th>
<th>CODE OFFSET</th>
<th>STATEMENT LABEL</th>
<th>CODE OFFSET</th>
<th>STATEMENT LABEL</th>
<th>CODE OFFSET</th>
<th>STATEMENT LABEL</th>
<th>CODE OFFSET</th>
<th>STATEMENT LABEL</th>
<th>CODE OFFSET</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 FAT</td>
<td>0</td>
<td>2</td>
<td>117</td>
<td>3</td>
<td>241</td>
<td>11</td>
<td>150</td>
<td>12</td>
<td>457</td>
</tr>
</tbody>
</table>

Program Unit: HEARRANGSOURCE Compiled
```
00020 00046000 00046600 00046700 00046800 00046900 00046A00 00046B00 00046C00 00046D00 00046E00 00046F00 00047000 00047100 00047200 00047300 00047400 00047500 00047600 00047700 00047800 00047900 00047A00 00047B00 00047C00 00047D00 00047E00 00047F00 00048000 00048100 00048200 00048300 00048400 00048500 00048600 00048700 00048800 00048900 00048A00 00048B00 00048C00 00048D00 00048E00 00048F00 00049000 00049100 00049200 00049300 00049400 00049500 00049600 00049700 00049800 00049900 00049A00 00049B00 00049C00 00049D00 00049E00 00049F00 0004A000 0004A100 0004A200 0004A300 0004A400 0004A500 0004A600 0004A700 0004A800 0004A900 0004AA00 0004AB00 0004AC00 0004AD00 0004AE00 0004AF00 0004B000 0004B100 0004B200 0004B300 0004B400 0004B500 0004B600 0004B700 0004B800 0004B900 0004BA00 0004BB00 0004BC00 0004BD00 0004BE00 0004BF00 0004C000 0004C100 0004C200 0004C300 0004C400 0004C500 0004C600 0004C700 0004C800 0004C900 0004CA00 0004CB00 0004CC00 0004CD00 0004CE00 0004CF00 0004D000 0004D100 0004D200 0004D300 0004D400 0004D500 0004D600 0004D700 0004D800 0004D900 0004DA00 0004DB00 0004DC00 0004DD00 0004DE00 0004DF00 0004E000 0004E100 0004E200 0004E300 0004E400 0004E500 0004E600 0004E700 0004E800 0004E900 0004EA00 0004EB00 0004EC00 0004ED00 0004EE00 0004EF00 0004F000 0004F100 0004F200 0004F300 0004F400 0004F500 0004F600 0004F700 0004F800 0004F900 0004FA00 0004FB00 0004FC00 0004FD00 0004FE00 0004FF00

SYMBOL MAP

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>STRUCTURE</th>
<th>ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASTRING</td>
<td>CHARAC</td>
<td>SIMPLE VAN 0-9Z, A</td>
<td>1</td>
</tr>
<tr>
<td>INDETSUBNAME</td>
<td>STRING</td>
<td>SIMPLE VAN 0-9, A</td>
<td>1</td>
</tr>
<tr>
<td>LINECOUNTER</td>
<td>INTEGER</td>
<td>SIMPLE VAN 0-9N</td>
<td>1</td>
</tr>
<tr>
<td>STRING</td>
<td>CHARAC</td>
<td>SIMPLE VAN 0-9, A</td>
<td>1</td>
</tr>
</tbody>
</table>

LABEL MAP

<table>
<thead>
<tr>
<th>STATEMENT CODE</th>
<th>STATEMENT CODE</th>
<th>STATEMENT CODE</th>
<th>STATEMENT CODE</th>
<th>STATEMENT CODE</th>
<th>STATEMENT CODE</th>
<th>STATEMENT CODE</th>
<th>STATEMENT CODE</th>
<th>STATEMENT CODE</th>
<th>STATEMENT CODE</th>
<th>STATEMENT CODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10</td>
<td>98</td>
<td>150</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

PROGRAM UNIT INDETSUBNAME COMPILED
```
0002 00840000 $CONTROL_INIT, MAP, LABEL, LOCATION
0005 00665000  SUBROUTINE PRINTLINE(0000, LINECOUNTER, PAGENUM, SOURCEFILE, STRING, LC)
0002 00840000  I$TPWN, TUTFILN, X$ECMB, RTUMCUMNTS, I$TPWMTS, TUTFILC)
0002 00667000  ZNUMCUMNTS, I$TPWMTS, NUCMBR, X$ECMB, I$TPWN, TUTFILC)
0002 00450000  CHARACT$ STRING(72, SOURCEFILE, PAGE0UM, X$ECMB)
0002 00590000  INTEGER RCMBGN, I$TPWN, PAGE0UM(50), PAGE0UM
0005 00840000  LINECOUNT$=LINECOUNTER
0002 00591000  IF(LINECOUNTER, GT, 0.0) CALL PAGE0UM(LINECOUNTER, PAGE0UM, SOURCEFILE)
0006 00492000  C DETERMINE IF STATISTICS HAVE ALREADY BEEN PRINTED FOR THIS PROGRAM.
0006 00591000  C IF TRUE DO NOT PRINT.
0006 00599000  IF(FLG0UMX(LC), L.W. 0, GUM TO 1)
0007 00859000  WRITE(*,6,66)&STRING(1521, STATPH1)
0117 00097000  66 FORMAT(A25,19)
0117 00097000  C INCREMENT CALL COUNTER FOR THIS PROGRAM UNIT
0117 00089000  USED(LC)$=USED(XLC)
0013 00090000  READ(RDUCEDCLUDC), I$TPWN, RCMB0UM, IEND, NUM0UM
0014 00090000  WRITE(REDUCECLUDC(LC), 1) I$TPWN, IBEGIN, IEND, NUM0UM, USED(LC)
0221 00840000  RETURN 1
0227 00930000  14 USEDCLUDC=USEDCLUDC
0232 00930000  NUMCLUDC=REDUCEDCLUDC-NUMCLUDC
0233 00930000  READ(REDUCECLUDC, NUMCLUDC, NUMCLUDC)
0246 00930000  NEWCLUDC=READ(XCLUDC)
0251 00870000  C DETERMINED TO BE FIRST CALL FOR THIS PROGRAM UNIT. THEREFORE, PRINT
0251 00904000  C ALL RELEVANT STATISTICS
0251 00905000  WRITE(*,77) &STRING(1031, NEWCLUDC, NEWCLUDC, RCMB0UM, NUMCLUDC)
0251 00910000  DUMCLUDC, NUMCLUDC, NUMCLUDC, RCMB0UM
0317 00910000  71 FORMATT(19,19,112,19,112,111)
0317 00910000  C ACCUMULATE 'F' OF LINE TOTALS
0317 00910000  CALL ACCUMULATECLUDC(TULINES, TUTFILC, TUTFILC, NUMCLUDC, TUTFILC)
0317 00910000  ACCUMULATECLUDC(TULINES, TUTFILC, TUTFILC, NUMCLUDC, TUTFILC)
0330 00915000  C WRITE NEW BEGINNING AND ENDING LINE 'S' S OF LINES, AND CALL
0330 00916000  C Callback TO 'REDUCED' ARRAY
0330 00917000  READ(REDUCECLUDC(LC), I$TPWN, IBEGIN, IEND)
0330 00918000  WRITE(REDUCECLUDC(LC), I$TPWN, IBEGIN, IEND, NUMCLUDC, USEDCLUDC)
0421 00919000  1 FORMAT(A12,14)
0421 00920000  RETURN
0422 00921000  END

SIMDUC MAP

NAME STRUCTURE ADDRESS
ACCUMULATECLUDC SIMPLE 0042
BEGIN INTEGER 0042
LC INTEGER 0042
NUMCLUDC INTEGER 0042
REDUCEDCLUDC INTEGER 0042
NUMCLUDC INTEGER 0042
READCLUDC INTEGER 0042
PAGECLUDC INTEGER 0042
STCTRL INTEGER 0042
TUTFILC INTEGER 0042

NAME STRUCTURE ADDRESS
ASTRING CHARACTER SIMPLE VAN 0046
LC INTEGER SIMPLE VAN 0046
LINECLUDC INTEGER SIMPLE VAN 0046
NUMCLUDC INTEGER SIMPLE VAN 0046
NUMCLUDC INTEGER SIMPLE VAN 0046
PAGECLUDC SUBROUTINE
READCLUDC PRINTLINE
RCMBGN INTEGER SIMPLE VAN 0046
SOURCEFILE CHARACTER SIMPLE VAN 0046
TUTFILC CHARACTER SIMPLE VAN 0046

NAME STRUCTURE ADDRESS
ACCUMULATECLUDC SIMPLE VAN 0042
BEGIN SIMPLE VAN 0042
LC SIMPLE VAN 0042
NUMCLUDC SIMPLE VAN 0042
REDUCEDCLUDC SIMPLE VAN 0042
NUMCLUDC SIMPLE VAN 0042
READCLUDC SIMPLE VAN 0042
PAGECLUDC SIMPLE VAN 0042
STCTRL SIMPLE VAN 0042
TUTFILC SIMPLE VAN 0042

NAME STRUCTURE ADDRESS
ASTRING CHARACTER SIMPLE VAN 0046
LC INTEGER SIMPLE VAN 0046
LINECLUDC INTEGER SIMPLE VAN 0046
NUMCLUDC INTEGER SIMPLE VAN 0046
NUMCLUDC INTEGER SIMPLE VAN 0046
PAGECLUDC SUBROUTINE
READCLUDC PRINTLINE
RCMBGN INTEGER SIMPLE VAN 0046
SOURCEFILE CHARACTER SIMPLE VAN 0046
TUTFILC CHARACTER SIMPLE VAN 0046
### SYMBOL MAP

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>STRUCTURE</th>
<th>ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCUMLINETOTALS</td>
<td></td>
<td>SUBROUTINE</td>
<td></td>
</tr>
<tr>
<td>NUMDECL</td>
<td>INTEGER</td>
<td>SIMPLE VAT</td>
<td>0-45 , 1</td>
</tr>
<tr>
<td>NUMLINES</td>
<td>INTEGER</td>
<td>SIMPLE VAT</td>
<td>0-610 , 1</td>
</tr>
<tr>
<td>TOTDECL</td>
<td>REAL</td>
<td>SIMPLE VAT</td>
<td>0-54 , 1</td>
</tr>
<tr>
<td>TOTLINES</td>
<td>REAL</td>
<td>SIMPLE VAT</td>
<td>0-618 , 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>STRUCTURE</th>
<th>ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMCOMMENTS</td>
<td>INTEGER</td>
<td>SIMPLE VAT</td>
<td>U-51 , 1</td>
</tr>
<tr>
<td>NUMFURMATS</td>
<td>INTEGER</td>
<td>SIMPLE VAT</td>
<td>U-56 , 1</td>
</tr>
<tr>
<td>TOTCOMMENTS</td>
<td>REAL</td>
<td>SIMPLE VAT</td>
<td>U-112 , 1</td>
</tr>
<tr>
<td>TOTFURMATS</td>
<td>REAL</td>
<td>SIMPLE VAT</td>
<td>U-111 , 1</td>
</tr>
</tbody>
</table>

PROGRAM UNIT ACCUMLINETOTALS COMPILED
00040 00931000 DCONTROL INIT,MAP,LABEL,LOCATIONS
00040 00931200 GCONTROL WRITEUNITFILE(LEADING,SECLMD,MCC)
00040 00931400 INTEGER SEGLEN,SECLMD
00040 00931600 CHARACTER STRING,LENGTH
00040 00931800 C WRITE PROGRAM UNIT TO FILE
00040 00932000 DO 30 IF(SEGLMD,SECLMD)
00040 00932200 READ(*1,1)STRING
00040 00932400 IF(30,31,32)
00040 00932600 NLCM,MCC+1
00040 00932800 LINESUM=SUM(MCC,3)
00040 00933000 DO 50 K=1,4
00040 00933200 IF(LINESUM(K),MCC,K)
00040 00933400 LINESUM(K)=K
00040 00933600 50 CONTINUE
00040 00933800 51 WRITE(*1,4)STRING,LINESUM
00040 00934000 4 FORMAT(2,81,800)
00040 00934200 30 CONTINUE
00040 00934400 RETURN
00040 00934600 END

SYMBL MAP

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>STRUCTURAL ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>INTEGER</td>
<td>SIMPLE VAN U=1</td>
</tr>
<tr>
<td>LINESUM</td>
<td>CHARACTER</td>
<td>SIMPLE VAN U=1</td>
</tr>
<tr>
<td>LEADING</td>
<td>CHARACTER</td>
<td>SIMPLE VAN U=1</td>
</tr>
<tr>
<td>SECLMD</td>
<td>INTEGER</td>
<td>SIMPLE VAN U=1</td>
</tr>
<tr>
<td>STRING</td>
<td>CHARACTER</td>
<td>SIMPLE VAN U=1</td>
</tr>
</tbody>
</table>

LABEL MAP

<table>
<thead>
<tr>
<th>STATEMENT</th>
<th>CODE</th>
<th>STATEMENT</th>
<th>CODE</th>
<th>STATEMENT</th>
<th>CODE</th>
<th>STATEMENT</th>
<th>CODE</th>
<th>STATEMENT</th>
<th>CODE</th>
<th>STATEMENT</th>
<th>CODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>LABEL</td>
<td>OFFSET</td>
<td>LABEL</td>
<td>OFFSET</td>
<td>LABEL</td>
<td>OFFSET</td>
<td>LABEL</td>
<td>OFFSET</td>
<td>LABEL</td>
<td>OFFSET</td>
<td>LABEL</td>
<td>OFFSET</td>
</tr>
<tr>
<td>1 FAT</td>
<td>0</td>
<td>4 FAT</td>
<td>5</td>
<td>30</td>
<td>174</td>
<td>50</td>
<td>146</td>
<td>51</td>
<td>147</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

PROGRAM UNIT WRITEUNITFILE COMPILED
00052  00960000  CONTROL INIT.MAP, LABEL, LOCATION
00053  00951000  SUBROUTINE DETERMINE CALL (STRING, CALL, NAME, ANGSUBSTITUTION, 
00054  00952000  IF, STRING, LEVEL, RECURSIVE, PROCURIT, HLC, DU, FUNCTION, LC, UNITNAME)
00055  00953000  CHARACTE R, SUBR1, CALL1, STRING1, ANGSUBSTITUTION1, 1, 8,
00056  00954000  LC1, ACTUALNAME1, FUNCTION1, NAME1, RECURSIVE1, PROCURIT1, HLC1
00057  00955000  INTEGER J, STRING, LEVEL, LC, HLC
00058  00956000  JSTART
00059  00958000  C STAGE "REDUCED" ANKAT'S ADDRESS OF CURRENT CALL STATEMENT
00060  00959000  LEVEL, LC1, HLC1
00061  00960000  LOCALLNAME, LEVEL, LC
00062  00961000  C CHECK FOR "ANGSUBSTITUTION" POINTER
00063  00962000  IF, CALL
00064  00963000  C DETERMINE PROCURIT CALLED
00065  00964000  IF, NAME=REDUCED, STRING, CALL, DU
00066  00965000  CALL, RCURRENTNAME, (HLC1, NAME, OVERPRINT)
00067  00966000  C CHECK FOR RECURSIVE CALL
00068  00967000  IF, NAME=LASTNAME, LC1 TO 44
00069  00968000  STRING(1:14), "RECURSIVE CALL"
00070  00969000  RETURN
00071  00970000  C CHECK FOR NAME SUBSTITUTION VIA USE OF "EXTERNAL" STATEMENT
00072  00971000  44 \STRING(1:14), "EXTERNAL VIA EXTERNAL"  DU, RETURN
00073  00972000  C FIND ACTUAL SUBROUTINE NAME IN "ANGSUBSTITUTION" ANKAT
00074  00973000  HLC1 TO 14, 8
00075  00974000  READ(ANGSUBSTITUTION, [1, 8], ACTUALNAME, FUNCTION, HLC1)
00076  00975000  IF, NAME=LASTNAME, LC1 TO 44
00077  00976000  CONTINUE
00078  00977000  C PROCURIT, BEGIN, UNNAMED, DU, RETURN
00079  00978000  IF, NAME=EXTERNAL, STRING, 11, RETURN
00080  00979000  C PROCURIT, BEGIN, UNNAMED, DU, RETURN
00081  00980000  PROGRIT3(1:11) TO 84
00082  00981000  PROGRIT3(2:11) TO FUNCTION
00083  00982000  CALL, FNPRGRIT, HLC1, DU, STRING, 11, 12, 14, RETURN, HLC1
00084  00983000  ILJC, SUB, FUNCTION, STRING, DU, HLC1, (1UM, 199)
00085  00984000  C SUBSTITUTE ACTUAL NAME FOR FORMAL NAME
00086  00985000  111 NAMEACTUALNAME
00087  00986000  RETURN
00088  00987000  C PROGRAM UNIT NOT FOUND: GET "ANGSUBSTITUTION" POINTER FROM
00089  00988000  C CALLING PROGRAM UNIT
00090  00989000  HLC1 TO 14, 8
00091  00990000  READ(REDUCED, LOCALLNAME, [1, 14])
00092  00991000  IF, NAME=REDUCED, STRING, HLC1, 11, RETURN
00093  00992000  J1=1
00094  00993000  LOCALLNAME, LEVEL, LC1
00095  00994000  GO TO 112
00096  00995000  98 DISPLAY 'HU PROGRAM UNIT FOUND FOR "NAME," PROGRAM IS STOPPED'
00097  00996000  STOP
00098  00997000  END

SYMBOL MAP

NAME TYPE STRUCTURE ADDRESS
ACTUALNAME CHARACTER SIMPLE VAR 0-114, 1
CALL CHARACTER SIMPLE VAR 0-121, 1
ANGSUBSTITUTION CHARACTER ANKAT 0-117, 1
SUBROUTINE
### Label Map

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>240</td>
<td>11</td>
<td>11</td>
<td>244</td>
<td>11</td>
<td>44</td>
<td>113</td>
<td>5</td>
<td>114</td>
<td>9</td>
<td>34</td>
</tr>
<tr>
<td>111</td>
<td>340</td>
<td>112</td>
<td>341</td>
<td>360</td>
<td>717</td>
<td>59</td>
<td>408</td>
<td>99</td>
<td>365</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Program Unit: DETERPHUGOCALL Compiled**
```
00047 00999000  $CONTROL INIT, MAP, LABEL, LOCATION
00047 00999000  SUBROUTINE PRINTTOTALS(TOTALINES, TOTCOMMENTS, TOTFORMATS, TOTDEBL)
00047 01000000  (NPRINTUNITS)
00047 01001000  TOTALINES=TOTCOMMENTS=TOTFORMATS=TOTDEBL
00047 01002000  &BIT66, 1)NPRINTUNITS,TOTALINES,TOTCOMMENTS,TOTFORMATS,TOTDEBL
00047 01003000  RETURN
00101 01004000  RETURN
00104 01005000  END

SYMBOL MAP

NAME      TYPE   STRUCTURE     ADDRESS  NAME      TYPE   STRUCTURE     ADDRESS
---------  ------  -----------  --------  ---------  ------  -----------  --------
NPRINTUNITS    INTEGER  SIMPLE VA1  0-45, 1  PRINTTOTALS    REAL   SIMPLE VA1  0-50, 1
TOTCOMMENTS    REAL   SIMPLE VA1  0-47, 1  TOTDEBL       REAL   SIMPLE VA1  0-51, 1
TOTFORMATS    REAL   SIMPLE VA1  0-46, 1  TOTLINE       REAL   SIMPLE VA1  0-48, 1
TOTDEBL       REAL   SIMPLE VA1  0-49, 1

LABEL MAP

STATEMENT CODE STATEMENT CODE STATEMENT CODE STATEMENT CODE STATEMENT CODE STATEMENT CODE STATEMENT CODE STATEMENT CODE
LABEL OFFSET  LABEL OFFSET  LABEL OFFSET  LABEL OFFSET  LABEL OFFSET  LABEL OFFSET  LABEL OFFSET  LABEL OFFSET

1 FAT 0

PROGRAM UNIT: PRINTTOTALS COMPILED
```
00010  01007000  SCPE
00010  01000000  CONTROL INIT, MAP, LABEL, LOCAL
00010  01000000  SYSTEM INTRINSIC FNN, FCLOSE
00010  01010000  NUMINES=NUMINES
00010  01010000  DETERMINE IF # OF OUTPUT LINES = # OF INPUT LINES
00010  01010000  IF(NUMINES.NE.NUMINES)GO TO 77
00010  01010000  DISPLAY**
00014  01014000  DISPLAY**
00014  01014000  C DISPLAY ERROR MESSAGE BECAUSE INPUT LINES DO NOT EQUAL OUTPUT LINES
00014  01014000  C AND TERMINATE PROGRAM WITHOUT CREATING NEW FILE.
00014  01014000  C AND TERMINATE PROGRAM WITHOUT CREATING NEW FILE.
00014  01014000  C AND TERMINATE PROGRAM WITHOUT CREATING NEW FILE.
00014  01014000  C AND TERMINATE PROGRAM WITHOUT CREATING NEW FILE.
00014  01014000  C AND TERMINATE PROGRAM WITHOUT CREATING NEW FILE.
00014  01014000  C AND TERMINATE PROGRAM WITHOUT CREATING NEW FILE.
00014  01014000  C AND TERMINATE PROGRAM WITHOUT CREATING NEW FILE.
00014  01014000  C AND TERMINATE PROGRAM WITHOUT CREATING NEW FILE.
00014  01014000  C AND TERMINATE PROGRAM WITHOUT CREATING NEW FILE.
00014  01014000  C AND TERMINATE PROGRAM WITHOUT CREATING NEW FILE.
00014  01014000  C AND TERMINATE PROGRAM WITHOUT CREATING NEW FILE.
00014  01014000  C AND TERMINATE PROGRAM WITHOUT CREATING NEW FILE.
00014  01014000  C AND TERMINATE PROGRAM WITHOUT CREATING NEW FILE.
00014  01014000  C AND TERMINATE PROGRAM WITHOUT CREATING NEW FILE.
00014  01014000  C AND TERMINATE PROGRAM WITHOUT CREATING NEW FILE.
00014  01014000  C AND TERMINATE PROGRAM WITHOUT CREATING NEW FILE.
00014  01014000  C AND TERMINATE PROGRAM WITHOUT CREATING NEW FILE.
00014  01014000  C AND TERMINATE PROGRAM WITHOUT CREATING NEW FILE.
00014  01014000  C AND TERMINATE PROGRAM WITHOUT CREATING NEW FILE.
00014  01014000  C AND TERMINATE PROGRAM WITHOUT CREATING NEW FILE.
00014  01014000  C AND TERMINATE PROGRAM WITHOUT CREATING NEW FILE.
00014  01014000  C AND TERMINATE PROGRAM WITHOUT CREATING NEW FILE.
00014  01014000  C AND TERMINATE PROGRAM WITHOUT CREATING NEW FILE.
00014  01014000  C AND TERMINATE PROGRAM WITHOUT CREATING NEW FILE.
00014  01014000  C AND TERMINATE PROGRAM WITHOUT CREATING NEW FILE.
00014  01014000  C AND TERMINATE PROGRAM WITHOUT CREATING NEW FILE.
00014  01014000  C AND TERMINATE PROGRAM WITHOUT CREATING NEW FILE.
00014  01014000  C AND TERMINATE PROGRAM WITHOUT CREATING NEW FILE.
00014  01014000  C AND TERMINATE PROGRAM WITHOUT CREATING NEW FILE.
00014  01014000  C AND TERMINATE PROGRAM WITHOUT CREATING NEW FILE.
00014  01014000  C AND TERMINATE PROGRAM WITHOUT CREATING NEW FILE.
00014  01014000  C AND TERMINATE PROGRAM WITHOUT CREATING NEW FILE.
FUNCTION CHARACTE $| XVISION + | U/V/1| "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPLE VAR | U/V/1 | "$ | 0| INTEGRAL NAME | SIMPL
LABEL MAP

<table>
<thead>
<tr>
<th>STATEMENT</th>
<th>CODE</th>
<th>STATEMENT</th>
<th>CODE</th>
<th>STATEMENT</th>
<th>CODE</th>
<th>STATEMENT</th>
<th>CODE</th>
<th>STATEMENT</th>
<th>CODE</th>
<th>STATEMENT</th>
<th>CODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>LABEL</td>
<td>OFFSET</td>
<td>LABEL</td>
<td>OFFSET</td>
<td>LABEL</td>
<td>OFFSET</td>
<td>LABEL</td>
<td>OFFSET</td>
<td>LABEL</td>
<td>OFFSET</td>
<td>LABEL</td>
<td>OFFSET</td>
</tr>
<tr>
<td>9 FAT</td>
<td>0</td>
<td>19 FAT</td>
<td>5</td>
<td>20</td>
<td>407</td>
<td>11 FAT</td>
<td>15</td>
<td>100</td>
<td>550</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

PROGRAM UNIT LISTNAMES COMPILED
PROGRAM UNIT PAGEHEAD COMPILED
I. INTRODUCTION

This discussion assumes that the object code of the FORTRAN source code rearranger program named 'REARRANG', is accessible to the user.

Before using this program to analyze and rearrange the source code file of a FORTRAN program, the following conditions must be met:

(1) all modules of the FORTRAN program must exist in one file. The name of this source code file can be from one to eight characters where the first character must be alphabetical.

(2) each module of the program should have been successfully compiled.

(3) all compiled modules of the program should have been successfully linked-loaded.

(4) the size of the program's source code cannot exceed 4000 lines.

(5) the number of subroutines and the number of functions must not exceed 100. The sum of functions plus subroutines cannot exceed 150.

(6) the total number of (CALL + SUBROUTINE + FUNCTION) statements cannot exceed 370.
II. EXECUTING THE PROGRAM

To execute the source code rearranger program, simply type 'RUN REARRANG'. The program will then ask for the name of the FORTRAN source code file to be analyzed. Respond by typing the name of the FORTRAN source code file. If the file name does not exist or if the file is not accessible, the program will terminate displaying the appropriate file system error. If the file name is valid and the file is accessible, no further user input is required.

If the code rearranger program executes successfully, it will actually overwrite the new rearranged file to the old source code file. This can be done because the file size does not change, but is simply rearranged. If program execution is unsuccessful, for any reason, the appropriate message is displayed and the original source code file is unchanged.

If the code rearranger is unsuccessful, it will be because of one of the following reasons:

(1) Duplicate module names have been detected. The name(s) and locations in the unchanged source code file are shown and the program is terminated.

(2) One or more modules exist in the source code but are not called by at least one other module. The name(s) and locations of the extraneous modules in the unchanged source code file are shown and the program is terminated.
(Although presence of not used modules does not prevent successful execution of a program, this detection does serve as a useful housekeeping and file space conservation utility.)

The user can then edit the source code file based on the above error detection information that has been given. Then, the code rearranger program can be executed again.

When the code rearranger program is successful, the source code file is rearranged concurrent with the outputting of the first of two reports, which shows the hierarchical top-down order of the modules of the FORTRAN source code. Included in this report are the beginning and ending line number in the newly rearranged file for each module. Also, for each module, the level of its call, the total number, number of comment, declaration, format, and executable lines of code are shown.

A second report is then output showing all the modules listed in alphabetical order with beginning and ending line number and number of times called for each module. Also, this report shows the number of lines for the largest, smallest, and average module size and the percent of lines of code classified as comment, declaration, format, and executable.

Upon successful execution of this program, it is suggested that a compiled listing of the rearranged source code file be generated. Then the output of the code rearranger program filed with this compiled listing will serve as both
a table of contents and an index for the FORTRAN program.

Whenever any program maintenance is done, the code rearranger program should be run for the affected FORTRAN source code and a new compiled listing generated.
APPENDIX C

EXAMPLES OF ANALYZED PROGRAMS
<table>
<thead>
<tr>
<th>PROGRAM UNIT NAME</th>
<th>BEG. LINE</th>
<th>END. LINE</th>
<th>LEVEL</th>
<th>OF LINES</th>
<th>OF COMM.</th>
<th>OF DECLAR.</th>
<th>OF FORMATS</th>
<th>OF XEG LINES</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAIN</td>
<td>1</td>
<td>59</td>
<td>1</td>
<td>59</td>
<td>19</td>
<td>6</td>
<td>0</td>
<td>34</td>
</tr>
<tr>
<td>SUBROUTINE DEFINEFILES</td>
<td>60</td>
<td>90</td>
<td>2</td>
<td>31</td>
<td>9</td>
<td>3</td>
<td>3</td>
<td>16</td>
</tr>
<tr>
<td>SUBROUTINE FINDSOURCEBEGIN</td>
<td>91</td>
<td>132</td>
<td>2</td>
<td>42</td>
<td>4</td>
<td>7</td>
<td>2</td>
<td>27</td>
</tr>
<tr>
<td>SUBROUTINE STRIPOFFNAME</td>
<td>133</td>
<td>141</td>
<td>3</td>
<td>29</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>24</td>
</tr>
<tr>
<td>SUBROUTINE STORENAME</td>
<td>142</td>
<td>171</td>
<td>3</td>
<td>10</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>SUBROUTINE DETERMINENAME</td>
<td>172</td>
<td>178</td>
<td>4</td>
<td>7</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>SUBROUTINE STORENAME</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE DETERMINENAME</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE CLASSIFIERSTATEMENT</td>
<td>179</td>
<td>234</td>
<td>2</td>
<td>56</td>
<td>10</td>
<td>7</td>
<td>1</td>
<td>30</td>
</tr>
<tr>
<td>SUBROUTINE DETERMINDISPUT</td>
<td>235</td>
<td>250</td>
<td>3</td>
<td>16</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>SUBROUTINE PROCESSCALL</td>
<td>251</td>
<td>277</td>
<td>3</td>
<td>27</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>SUBROUTINE PROCESSIF</td>
<td>278</td>
<td>293</td>
<td>4</td>
<td>16</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>13</td>
</tr>
<tr>
<td>SUBROUTINE STRIPOFFNAME</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE CHECKARGUMENT</td>
<td>294</td>
<td>348</td>
<td>3</td>
<td>55</td>
<td>17</td>
<td>5</td>
<td>2</td>
<td>31</td>
</tr>
<tr>
<td>SUBROUTINE DETERMINDISPUT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE DETERMINDISPUT</td>
<td>349</td>
<td>390</td>
<td>3</td>
<td>42</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>36</td>
</tr>
<tr>
<td>SUBROUTINE DETERMINDISPUT</td>
<td>391</td>
<td>429</td>
<td>3</td>
<td>39</td>
<td>6</td>
<td>6</td>
<td>1</td>
<td>26</td>
</tr>
<tr>
<td>SUBROUTINE DETERMINDISPUT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE DETERMINDISPUT</td>
<td>430</td>
<td>490</td>
<td>2</td>
<td>61</td>
<td>15</td>
<td>6</td>
<td>2</td>
<td>38</td>
</tr>
<tr>
<td>SUBROUTINE DETERMINDISPUT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE DETERMINDISPUT</td>
<td>491</td>
<td>504</td>
<td>3</td>
<td>14</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>11</td>
</tr>
<tr>
<td>SUBROUTINE DETERMINDISPUT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE DETERMINDISPUT</td>
<td>505</td>
<td>512</td>
<td>4</td>
<td>9</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>SUBROUTINE DETERMINDISPUT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE DETERMINDISPUT</td>
<td>513</td>
<td>544</td>
<td>2</td>
<td>52</td>
<td>10</td>
<td>5</td>
<td>4</td>
<td>33</td>
</tr>
<tr>
<td>SUBROUTINE DETERMINDISPUT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE DETERMINDISPUT</td>
<td>555</td>
<td>612</td>
<td>3</td>
<td>48</td>
<td>9</td>
<td>5</td>
<td>3</td>
<td>31</td>
</tr>
<tr>
<td>SUBROUTINE DETERMINDISPUT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE DETERMINDISPUT</td>
<td>613</td>
<td>651</td>
<td>2</td>
<td>37</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>29</td>
</tr>
<tr>
<td>SUBROUTINE DETERMINDISPUT</td>
<td>652</td>
<td>659</td>
<td>3</td>
<td>8</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>SUBROUTINE DETERMINDISPUT</td>
<td>669</td>
<td>675</td>
<td>3</td>
<td>16</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>SUBROUTINE DETERMINDISPUT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE DETERMINDISPUT</td>
<td>676</td>
<td>725</td>
<td>2</td>
<td>50</td>
<td>4</td>
<td>7</td>
<td>1</td>
<td>36</td>
</tr>
<tr>
<td>SUBROUTINE DETERMINDISPUT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE DETERMINDISPUT</td>
<td>726</td>
<td>758</td>
<td>3</td>
<td>33</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>25</td>
</tr>
<tr>
<td>SUBROUTINE DETERMINDISPUT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE DETERMINDISPUT</td>
<td>759</td>
<td>817</td>
<td>2</td>
<td>59</td>
<td>15</td>
<td>8</td>
<td>1</td>
<td>35</td>
</tr>
<tr>
<td>SUBROUTINE DETERMINDISPUT</td>
<td>818</td>
<td>840</td>
<td>3</td>
<td>23</td>
<td>0</td>
<td>3</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>SUBROUTINE DETERMINDISPUT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE DETERMINDISPUT</td>
<td>841</td>
<td>855</td>
<td>3</td>
<td>25</td>
<td>3</td>
<td>6</td>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>SUBROUTINE DETERMINDISPUT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE DETERMINDISPUT</td>
<td>855</td>
<td>893</td>
<td>3</td>
<td>18</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>13</td>
</tr>
<tr>
<td>SUBROUTINE DETERMINDISPUT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE DETERMINDISPUT</td>
<td>884</td>
<td>921</td>
<td>3</td>
<td>38</td>
<td>8</td>
<td>5</td>
<td>3</td>
<td>22</td>
</tr>
<tr>
<td>SUBROUTINE DETERMINDISPUT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE DETERMINDISPUT</td>
<td>922</td>
<td>930</td>
<td>4</td>
<td>9</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>PROGRAM UNIT NAME</td>
<td>BEGIN LINE</td>
<td>END LINE</td>
<td>LEVEL</td>
<td># OF LINES</td>
<td># OF COMM.</td>
<td># OF DECLAR.</td>
<td># OF FORMATS</td>
<td># OF XER LINES</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------</td>
<td>----------</td>
<td>-------</td>
<td>------------</td>
<td>------------</td>
<td>--------------</td>
<td>--------------</td>
<td>---------------</td>
</tr>
<tr>
<td>SUBROUTINE WRITEONFILE</td>
<td>931</td>
<td>949</td>
<td>3</td>
<td>19</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>13</td>
</tr>
<tr>
<td>SUBROUTINE DETERMFILECALL</td>
<td>950</td>
<td>997</td>
<td>3</td>
<td>48</td>
<td>9</td>
<td>6</td>
<td>2</td>
<td>31</td>
</tr>
<tr>
<td>SUBROUTINE DETERMINENAME</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE FINDFILEUNIT</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE PRINTTOTALS</td>
<td>999</td>
<td>1006</td>
<td>3</td>
<td>9</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>SUBROUTINE SAVEFILE</td>
<td>1007</td>
<td>1028</td>
<td>3</td>
<td>22</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>SUBROUTINE CONVFSTATISTIXIC</td>
<td>1029</td>
<td>1057</td>
<td>2</td>
<td>27</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>21</td>
</tr>
<tr>
<td>SUBROUTINE LISTNAMES</td>
<td>1058</td>
<td>1101</td>
<td>3</td>
<td>44</td>
<td>6</td>
<td>5</td>
<td>3</td>
<td>30</td>
</tr>
<tr>
<td>SUBROUTINE SORTARRAY</td>
<td>1102</td>
<td>1123</td>
<td>4</td>
<td>22</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>16</td>
</tr>
<tr>
<td>SUBROUTINE SPLIT</td>
<td>1123</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE PAGEHEAD</td>
<td>1123</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TOTALS FOR 36 PROGRAM UNITS**

<p>|                  | 1123 | 182 | 144 | 44 | 733 |</p>
<table>
<thead>
<tr>
<th>PROGRAM UNIT NAME</th>
<th>DEG. LINE</th>
<th>END LINE</th>
<th># OF LINES</th>
<th># OF TIMES CALLED</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAIN</td>
<td>1</td>
<td>59</td>
<td>59</td>
<td>1</td>
</tr>
<tr>
<td>SUBROUTINE ACCUMULATETOTALS</td>
<td>922</td>
<td>930</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>SUBROUTINE CLASSIFYSTATEMENT</td>
<td>179</td>
<td>234</td>
<td>56</td>
<td>1</td>
</tr>
<tr>
<td>SUBROUTINE COMPLETEARGUMENT</td>
<td>513</td>
<td>564</td>
<td>52</td>
<td>1</td>
</tr>
<tr>
<td>SUBROUTINE COMPUTESTATISTIC</td>
<td>1629</td>
<td>1657</td>
<td>29</td>
<td>1</td>
</tr>
<tr>
<td>SUBROUTINE DEFINEFILES</td>
<td>60</td>
<td>90</td>
<td>31</td>
<td>1</td>
</tr>
<tr>
<td>SUBROUTINE DELETETLICALLS</td>
<td>391</td>
<td>429</td>
<td>39</td>
<td>1</td>
</tr>
<tr>
<td>SUBROUTINE DETERDECLARE</td>
<td>349</td>
<td>390</td>
<td>42</td>
<td>2</td>
</tr>
<tr>
<td>SUBROUTINE DETERDSGFM</td>
<td>235</td>
<td>250</td>
<td>16</td>
<td>2</td>
</tr>
<tr>
<td>SUBROUTINE DETERMINED</td>
<td>170</td>
<td>178</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>SUBROUTINE DETERMIFCALL</td>
<td>676</td>
<td>725</td>
<td>50</td>
<td>1</td>
</tr>
<tr>
<td>SUBROUTINE DETERPLECTION</td>
<td>950</td>
<td>997</td>
<td>48</td>
<td>1</td>
</tr>
<tr>
<td>SUBROUTINE DISPLAYPHASES</td>
<td>660</td>
<td>675</td>
<td>16</td>
<td>1</td>
</tr>
<tr>
<td>SUBROUTINE DISPLAYNOCALLS</td>
<td>726</td>
<td>758</td>
<td>33</td>
<td>2</td>
</tr>
<tr>
<td>SUBROUTINE FINDPROGRAMUNIT</td>
<td>841</td>
<td>865</td>
<td>25</td>
<td>2</td>
</tr>
<tr>
<td>SUBROUTINE FINDSAMEUNIT</td>
<td>613</td>
<td>651</td>
<td>39</td>
<td>1</td>
</tr>
<tr>
<td>SUBROUTINE FINDSAMEUNIT</td>
<td>98</td>
<td>132</td>
<td>42</td>
<td>1</td>
</tr>
<tr>
<td>SUBROUTINE GETSUBARNAME</td>
<td>565</td>
<td>612</td>
<td>48</td>
<td>1</td>
</tr>
<tr>
<td>SUBROUTINE INCENTSUBNAME</td>
<td>844</td>
<td>883</td>
<td>18</td>
<td>1</td>
</tr>
<tr>
<td>SUBROUTINE INSERTFUCALLS</td>
<td>430</td>
<td>490</td>
<td>61</td>
<td>1</td>
</tr>
<tr>
<td>SUBROUTINE LIFHTHAMES</td>
<td>1056</td>
<td>1101</td>
<td>44</td>
<td>1</td>
</tr>
<tr>
<td>SUBROUTINE PAGEHEAD</td>
<td>1102</td>
<td>1123</td>
<td>22</td>
<td>3</td>
</tr>
<tr>
<td>SUBROUTINE PAGEHEAD</td>
<td>818</td>
<td>840</td>
<td>23</td>
<td>2</td>
</tr>
<tr>
<td>SUBROUTINE PRINTLINE</td>
<td>804</td>
<td>921</td>
<td>38</td>
<td>1</td>
</tr>
<tr>
<td>SUBROUTINE PRINTTOTALS</td>
<td>998</td>
<td>1006</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>SUBROUTINE PROCESS</td>
<td>251</td>
<td>277</td>
<td>27</td>
<td>1</td>
</tr>
<tr>
<td>SUBROUTINE PROCESS</td>
<td>278</td>
<td>293</td>
<td>16</td>
<td>1</td>
</tr>
<tr>
<td>SUBROUTINE REARRANGEMENT</td>
<td>759</td>
<td>817</td>
<td>59</td>
<td>1</td>
</tr>
<tr>
<td>SUBROUTINE SAVENVFILE</td>
<td>1007</td>
<td>1026</td>
<td>22</td>
<td>1</td>
</tr>
<tr>
<td>SUBROUTINE SORTARRAY</td>
<td>491</td>
<td>504</td>
<td>14</td>
<td>3</td>
</tr>
<tr>
<td>SUBROUTINE STONEUNIT</td>
<td>162</td>
<td>171</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>SUBROUTINE GTPUFFUM</td>
<td>133</td>
<td>161</td>
<td>29</td>
<td>2</td>
</tr>
<tr>
<td>SUBROUTINE UTIC</td>
<td>452</td>
<td>459</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>SUBROUTINE UNICH</td>
<td>565</td>
<td>512</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>SUBROUTINE WRTETONEFILE</td>
<td>931</td>
<td>949</td>
<td>19</td>
<td>1</td>
</tr>
</tbody>
</table>

TOTALS FOR 36 PROGRAM UNITS = 1123  53

LARGEST PROGRAM UNIT SIZE = 61 LINES
SMALLEST PROGRAM UNIT SIZE = 7 LINES
AVERAGE PROGRAM UNIT SIZE = 31 LINES

% COMMENT LINES = 16
% DECLARATION LINES = 13
% FORMAT LINES = 4
% EXECUTABLE LINES = 67
<table>
<thead>
<tr>
<th>PROGRAM UNIT NAME</th>
<th>DEG. LINE</th>
<th>END LINE</th>
<th>LEVEL</th>
<th># OF LINES</th>
<th># OF COMM.</th>
<th># OF DECLAR.</th>
<th># OF FORMATS</th>
<th># OF EXEC. LINES</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAIN</td>
<td>1</td>
<td>83</td>
<td>1</td>
<td>83</td>
<td>22</td>
<td>14</td>
<td>0</td>
<td>47</td>
</tr>
<tr>
<td>SUBROUTINE READERFAR</td>
<td>84</td>
<td>106</td>
<td>2</td>
<td>23</td>
<td>0</td>
<td>5</td>
<td>1</td>
<td>17</td>
</tr>
<tr>
<td>SUBROUTINE READPRFAR</td>
<td>107</td>
<td>147</td>
<td>2</td>
<td>41</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>35</td>
</tr>
<tr>
<td>SUBROUTINE DEFINFILE</td>
<td>140</td>
<td>168</td>
<td>2</td>
<td>21</td>
<td>0</td>
<td>3</td>
<td>5</td>
<td>13</td>
</tr>
<tr>
<td>SUBROUTINE DATEDETERMINE</td>
<td>149</td>
<td>271</td>
<td>2</td>
<td>103</td>
<td>0</td>
<td>9</td>
<td>4</td>
<td>82</td>
</tr>
<tr>
<td>SUBROUTINE BSSITCH</td>
<td>272</td>
<td>279</td>
<td>3</td>
<td>8</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>SUBROUTINE TERENCEWEEK</td>
<td>280</td>
<td>322</td>
<td>3</td>
<td>43</td>
<td>5</td>
<td>4</td>
<td>1</td>
<td>33</td>
</tr>
<tr>
<td>SUBROUTINE ENTERWEEKNO</td>
<td>323</td>
<td>346</td>
<td>2</td>
<td>24</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>21</td>
</tr>
<tr>
<td>SUBROUTINE CHANGEDAYJ</td>
<td>347</td>
<td>377</td>
<td>2</td>
<td>31</td>
<td>3</td>
<td>4</td>
<td>0</td>
<td>24</td>
</tr>
<tr>
<td>SUBROUTINE TRUCKHOURS</td>
<td>378</td>
<td>396</td>
<td>2</td>
<td>19</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>14</td>
</tr>
<tr>
<td>SUBROUTINE NODALHOURS</td>
<td>397</td>
<td>415</td>
<td>2</td>
<td>19</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>14</td>
</tr>
<tr>
<td>SUBROUTINE READDATA</td>
<td>414</td>
<td>443</td>
<td>3</td>
<td>48</td>
<td>4</td>
<td>8</td>
<td>0</td>
<td>36</td>
</tr>
<tr>
<td>SUBROUTINE ADFORCHANGES</td>
<td>444</td>
<td>537</td>
<td>3</td>
<td>64</td>
<td>5</td>
<td>10</td>
<td>0</td>
<td>49</td>
</tr>
<tr>
<td>SUBROUTINE SHORTOTALS</td>
<td>528</td>
<td>560</td>
<td>4</td>
<td>33</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>26</td>
</tr>
<tr>
<td>SUBROUTINE ACCEPTCHANGES</td>
<td>561</td>
<td>624</td>
<td>4</td>
<td>44</td>
<td>1</td>
<td>7</td>
<td>0</td>
<td>56</td>
</tr>
<tr>
<td>SUBROUTINE SHORTOTALS</td>
<td>625</td>
<td>656</td>
<td>4</td>
<td>32</td>
<td>1</td>
<td>9</td>
<td>0</td>
<td>22</td>
</tr>
<tr>
<td>SUBROUTINE SHORTDATA</td>
<td>657</td>
<td>669</td>
<td>3</td>
<td>33</td>
<td>1</td>
<td>8</td>
<td>0</td>
<td>24</td>
</tr>
<tr>
<td>SUBROUTINE CHANGEDAYN</td>
<td>690</td>
<td>710</td>
<td>2</td>
<td>21</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td>SUBROUTINE GETH</td>
<td>711</td>
<td>759</td>
<td>3</td>
<td>49</td>
<td>2</td>
<td>9</td>
<td>2</td>
<td>364</td>
</tr>
<tr>
<td>SUBROUTINE LUMPDAYN</td>
<td>760</td>
<td>796</td>
<td>3</td>
<td>37</td>
<td>1</td>
<td>10</td>
<td>0</td>
<td>26</td>
</tr>
<tr>
<td>SUBROUTINE GETDAYN</td>
<td>797</td>
<td>824</td>
<td>3</td>
<td>28</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>19</td>
</tr>
<tr>
<td>SUBROUTINE CONCATENATE</td>
<td>825</td>
<td>834</td>
<td>3</td>
<td>10</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>SUBROUTINE CUMHOURS</td>
<td>835</td>
<td>920</td>
<td>4</td>
<td>86</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>67</td>
</tr>
<tr>
<td>SUBROUTINE CONCATENATE</td>
<td>921</td>
<td>964</td>
<td>3</td>
<td>44</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>32</td>
</tr>
<tr>
<td>SUBROUTINE FINDCOMPONENT</td>
<td>945</td>
<td>1008</td>
<td>4</td>
<td>44</td>
<td>5</td>
<td>10</td>
<td>0</td>
<td>29</td>
</tr>
<tr>
<td>SUBROUTINE CONCATENATE</td>
<td>1009</td>
<td>1040</td>
<td>5</td>
<td>32</td>
<td>3</td>
<td>7</td>
<td>0</td>
<td>22</td>
</tr>
<tr>
<td>SUBROUTINE CONCATENATE</td>
<td>1040</td>
<td>1045</td>
<td>6</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>14</td>
</tr>
<tr>
<td>SUBROUTINE COMFOOOTHFAC</td>
<td>1041</td>
<td>1061</td>
<td>4</td>
<td>21</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>42</td>
</tr>
<tr>
<td>SUBROUTINE COMPUTREC</td>
<td>1062</td>
<td>1114</td>
<td>2</td>
<td>53</td>
<td>4</td>
<td>6</td>
<td>1</td>
<td>42</td>
</tr>
<tr>
<td>SUBROUTINE SUBHANDLESPA</td>
<td>1115</td>
<td>1147</td>
<td>3</td>
<td>35</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>27</td>
</tr>
<tr>
<td>SUBROUTINE SWITCH</td>
<td>1150</td>
<td>1157</td>
<td>4</td>
<td>8</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>SUBROUTINE CONPROREC</td>
<td>1158</td>
<td>1165</td>
<td>2</td>
<td>28</td>
<td>3</td>
<td>10</td>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>SUBROUTINE DETERMINED</td>
<td>1166</td>
<td>1203</td>
<td>3</td>
<td>17</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>11</td>
</tr>
<tr>
<td>SUBROUTINE ACCUMUL</td>
<td>1203</td>
<td>1215</td>
<td>3</td>
<td>13</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>SUBROUTINE CUMHOURS</td>
<td>1216</td>
<td>1242</td>
<td>2</td>
<td>27</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>20</td>
</tr>
<tr>
<td>SUBROUTINE COMPUTSETUPHRS</td>
<td>1243</td>
<td>1270</td>
<td>2</td>
<td>28</td>
<td>4</td>
<td>9</td>
<td>0</td>
<td>15</td>
</tr>
</tbody>
</table>
### SUMMARY OF EXECUTION SEQUENCE OF PROGRAM UNITS FOR FORTRAN SOURCE CODE: RESULTS

<table>
<thead>
<tr>
<th>PROGRAM UNIT NAME</th>
<th>DEG. LINE</th>
<th>END LINE</th>
<th>LEVEL</th>
<th># OF OPER.</th>
<th># OF COM.</th>
<th># OF DECL.</th>
<th># OF FORMATS</th>
<th># OF XEQ LINES</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUBROUTINE BETHRMMKDO</td>
<td>1271</td>
<td>1300</td>
<td>2</td>
<td>30</td>
<td>3</td>
<td>10</td>
<td>0</td>
<td>17</td>
</tr>
<tr>
<td>SUBROUTINE COMPCHARGEHS</td>
<td>1301</td>
<td>1317</td>
<td>3</td>
<td>17</td>
<td>3</td>
<td>4</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>SUBROUTINE ACCCHARGEHS</td>
<td>1271</td>
<td>1300</td>
<td>2</td>
<td>30</td>
<td>3</td>
<td>10</td>
<td>0</td>
<td>17</td>
</tr>
<tr>
<td>SUBROUTINE STORDATA</td>
<td>1319</td>
<td>1420</td>
<td>2</td>
<td>111</td>
<td>4</td>
<td>11</td>
<td>25</td>
<td>71</td>
</tr>
<tr>
<td>SUBROUTINE OUTPUTEFF</td>
<td>1429</td>
<td>1482</td>
<td>3</td>
<td>54</td>
<td>2</td>
<td>10</td>
<td>0</td>
<td>42</td>
</tr>
<tr>
<td>SUBROUTINE PRFVEFF</td>
<td>1483</td>
<td>1497</td>
<td>3</td>
<td>15</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>SUBROUTINE QUITPEARN</td>
<td>1498</td>
<td>1613</td>
<td>2</td>
<td>116</td>
<td>4</td>
<td>14</td>
<td>14</td>
<td>84</td>
</tr>
<tr>
<td>SUBROUTINE GUMBAR</td>
<td>1614</td>
<td>1700</td>
<td>3</td>
<td>87</td>
<td>1</td>
<td>13</td>
<td>0</td>
<td>71</td>
</tr>
<tr>
<td>SUBROUTINE LNCHTPE1</td>
<td>1701</td>
<td>1713</td>
<td>3</td>
<td>13</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>SUBROUTINE LNCHTPE2</td>
<td>1714</td>
<td>1741</td>
<td>3</td>
<td>28</td>
<td>0</td>
<td>4</td>
<td>.1</td>
<td>23</td>
</tr>
<tr>
<td>TOTALS FOR 44 PROGRAM UNITS</td>
<td>1741</td>
<td>121</td>
<td>291</td>
<td>59</td>
<td>1270</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROGRAM UNIT NAME</td>
<td>D.E.G. LINE</td>
<td>E.N.D. LINE</td>
<td># OF LINES</td>
<td># OF TIMES CALLED</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>------------</td>
<td>------------</td>
<td>------------</td>
<td>------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAIN</td>
<td>1</td>
<td>83</td>
<td>83</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE ACCEPTCHANGES</td>
<td>541</td>
<td>624</td>
<td>44</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE ACCUMCHARGERS</td>
<td>1301</td>
<td>1317</td>
<td>17</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE ACCUMHOURS</td>
<td>1203</td>
<td>1215</td>
<td>13</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE AKFUNCHANGES</td>
<td>464</td>
<td>527</td>
<td>44</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE 5CHANGEAJ</td>
<td>347</td>
<td>377</td>
<td>31</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE 5CHANGB</td>
<td>490</td>
<td>710</td>
<td>21</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE CHECKFOROUTHPAC</td>
<td>10941</td>
<td>1061</td>
<td>21</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE COMPCHARGERS</td>
<td>1271</td>
<td>1306</td>
<td>30</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE COMPJOINCHG</td>
<td>1216</td>
<td>1242</td>
<td>27</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE COMPONENTHOURS</td>
<td>1009</td>
<td>1040</td>
<td>32</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE COMPREGEC</td>
<td>1158</td>
<td>1105</td>
<td>20</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE COMPUTENCEDATA</td>
<td>626</td>
<td>656</td>
<td>32</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE COMPUTEREC</td>
<td>1085</td>
<td>1014</td>
<td>53</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE COMPUTEREGTURRS</td>
<td>1243</td>
<td>1270</td>
<td>20</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE CONCATENATE</td>
<td>025</td>
<td>034</td>
<td>10</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE CUMHOURS</td>
<td>921</td>
<td>954</td>
<td>44</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE CUMHOURS31</td>
<td>885</td>
<td>920</td>
<td>96</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE DATEDETERMINE</td>
<td>149</td>
<td>271</td>
<td>103</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE DEFINEFILES</td>
<td>140</td>
<td>140</td>
<td>21</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE DETERLENCHG</td>
<td>200</td>
<td>222</td>
<td>44</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE DETERLENCHG3</td>
<td>1186</td>
<td>1262</td>
<td>17</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE ENTERRENSO</td>
<td>965</td>
<td>1000</td>
<td>44</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE FINDCOMPONENT</td>
<td>711</td>
<td>758</td>
<td>49</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE GETHMS</td>
<td>711</td>
<td>758</td>
<td>49</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE HEQIAN</td>
<td>797</td>
<td>824</td>
<td>28</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE LINETYPE1</td>
<td>1701</td>
<td>1713</td>
<td>13</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE LINETYPE2</td>
<td>1714</td>
<td>1741</td>
<td>20</td>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE LISPDATA</td>
<td>766</td>
<td>796</td>
<td>37</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE NORMALHOURS</td>
<td>377</td>
<td>415</td>
<td>19</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE OUTPUTERR</td>
<td>1498</td>
<td>1613</td>
<td>116</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE OUTPUTEFF</td>
<td>1318</td>
<td>1428</td>
<td>111</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE PRECISE</td>
<td>1403</td>
<td>1497</td>
<td>35</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE READDATA</td>
<td>416</td>
<td>463</td>
<td>48</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE READFILEPAR</td>
<td>107</td>
<td>147</td>
<td>41</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE READGRIDPAR</td>
<td>84</td>
<td>101</td>
<td>23</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE SHOWTOTALS</td>
<td>520</td>
<td>560</td>
<td>33</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE SORTANGDISPLAY</td>
<td>1115</td>
<td>1149</td>
<td>35</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE SUPPCHG</td>
<td>272</td>
<td>279</td>
<td>8</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE STOREDATA</td>
<td>657</td>
<td>689</td>
<td>33</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE SUN</td>
<td>1429</td>
<td>1482</td>
<td>54</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE SUMERR</td>
<td>1614</td>
<td>1700</td>
<td>87</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE SUMHOURS</td>
<td>1150</td>
<td>1157</td>
<td>6</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBROUTINE TRUCKHOURS</td>
<td>378</td>
<td>376</td>
<td>19</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TOTALS FOR 44 PROGRAM UNITS: 1741 85
<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Largest Program Unit Size</td>
<td>116 lines</td>
</tr>
<tr>
<td>Smallest Program Unit Size</td>
<td>0 lines</td>
</tr>
<tr>
<td>Average Program Unit Size</td>
<td>40 lines</td>
</tr>
<tr>
<td>% Comment Lines</td>
<td>7</td>
</tr>
<tr>
<td>% Declaration Lines</td>
<td>17</td>
</tr>
<tr>
<td>% Format Lines</td>
<td>3</td>
</tr>
<tr>
<td>% Executable Lines</td>
<td>73</td>
</tr>
</tbody>
</table>

End of Program
BIBLIOGRAPHY

1. Fosdick, L.D., 'BRNANL, a Fortran program to identify basic blocks in Fortran programs', Report CU-CS040-74 Department of CS, University of Colorado.


AN IMPLEMENTATION OF A FORTRAN SOURCE CODE REARRANGER AND DOCUMENTATION GENERATOR PROGRAM

by

ARTHUR SCOTT MYERS

B.S., Lehigh University, 1969

-------------------

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the requirements of the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1983
With the explosive growth in the size of software, the analysis and documentation of module relationships for programs written in high level programming languages are undoubtedly useful and needed capabilities. This paper describes an implementation project for processing the source code of FORTRAN programs which produces such an analysis and documentation. The system that resulted from this implementation analyzes and rearranges a complete FORTRAN source code file into its hierarchical top-down order of modules. This system also provides two reports, which contain relevant information about each module's structure and size. The first report effectively shows the hierarchical arrangement of modules and can be thought of as a table of contents for the just rearranged program; the second report lists alphabetically by name the program's modules and therefore is the index. The paper discusses the requirements for implementing this project, the design decisions, and the actual implementation. Future work and extensions pertaining to this project are also considered.