ALGORITHM DESIGN OF A COMPUTER AIDED INSTRUCTION PACKAGE FACILITATING THE INSTRUCTOR-STUDENT RELATIONSHIP

By

LEE MERLE COLAW

Bachelor of Science

Oklahoma State University

Stillwater, Oklahoma

1975

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 1985

ALGORITHM DESIGN OF A COMPUTER AIDED
INSTRUCTION PACKAGE FACILITATING
THE INSTRUCTOR-STUDENT
RELATIONSHIP

Thesis Approved:

Kelvin L. Davis
Thesis Adviser

D. E. Hedrick

Michael P. Khen [illegible]

Norman N. Durham
Dean of the Graduate College

ACKNOWLEDGEMENTS

I wish to express my appreciation and gratitude to the U.S. Army, who deemed me worthy of the privilege to attend graduate school through the Army's Fully-Funded Graduate School Program.

I thank my advisors Dr. Kelvin L. Davis, Dr. G.E. Hedrick, and Dr. Michael E. Kerr for their guidance, support, and friendship throughout my graduate endeavor.

I am indebted to the many faculty members which have provided assistance to my studies and made my graduate education a success. I thank Dr. Sharilyn A. Thoreson, Dr. Donald W. Grace, Dr. Donald D. Fisher, and Mr. Arlen N. Long for their help and constructive comments throughout my graduate program.

Finally, a special thanks to my wife and two sons, Cheryl, Christopher and Benjamin.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Background Information

Since the beginning of time, mankind has continuously sought to improve his personal knowledge of the surrounding world. But only during the recent years has there been a vast amount of educational tools in which to better facilitate the learning process. Even with all of the modern technology available there are still a vast number of problems facing the instructors of today.

As an example, look at a computer science class which may be concerned with learning several introductory programming languages. The study of different computer languages poses a unique instructor-student situation. On one hand, the instructor may have students who are studying their first computer programming language. These students are probably both anxious and apprehensive concerning the course and it's contents. On the other hand, some of the students within the computer programming class may be knowledgeable of the programming language being studied and the thought processes required to formulate a problem into a working program. This situation is further compounded by students who like the course of study and consider it beneficial for their future well being, compared to those who have no desire or personal motivation for the subject and are undertaking the course only to meet some stated requirement. Class sizes are usually large which means that individual attention is minimal. Furthermore, class discussions are usually perceived by the less able students as no more than dialogues with the more advanced and exceptional students. Instructors are further plagued by demands upon their time to publish, keep abreast of current and emerging industry trends, and class preparation time. This scenario is specifically related to an environment within the computer science arena, however it's similarities with other courses should be evident.

Current modes of instructional presentation begins with the textbook. Most course textbooks are targeted for particular audiences based upon the student's assumed previous background. In addition, most textbooks are not 100% compatible with the current trends or local facilities. Introductory level textbooks are usually boring or difficult, even to the most dedicated of students. The use of various types of audiovisual aids is usually limited to pictures in the textbook, the chalk-board, and/or the overhead projector. Other teaching devices exist in the form of video cassettes, films and educational television. Each of these forms of media are either difficult to produce and modify, or they incur significant expense. But, perhaps one of the best and most flexible tools to come to the aid of a instructor is the computer.

The computer provides the instructor with the means to disseminate knowledge at an accelerated rate to those who need or desire the information thus freeing classroom time to concentrate on specific applications and emerging technical theory. Information accessed through a computer terminal by a student can be presented by a computer aided instruction (CAI) module or by a computer managed instruction (CMI) module. A CAI module consists of lesson objectives presented to the student through a terminal, with a text orientation usually followed by an on-line test based on the previous instructional information. Whereas CMI consists of various instructional blocks which may or may not contain exercises which supports an instructor's already established objectives, tests, and other forms of presentation materials external to the displays of the computer terminal (Dimas 1978; Spitler and Corgan 1979). CMI is considered to be a prerequisite of CAI learning. Either way, through the use of CAI or CMI, the computer is seen as a new means of communication for instructors to deliver course assignments and to evaluate student's responses concerning those assignments. In this type of environment, a student would proceed through assignments in an interactive mode which would require the computer to evaluate the responses.

Nature of the Problem

This study is concerned with the development of an algorithm package designed to facilitate computer aided instruction assignment presentation and interactive response evaluation. Presently, there are only a few commercially produced computer aided instruction packages obtainable by educators. The cost of such products is usually high and the quality of its content and ease of use is dependent upon its user. In response to this situation and the continuing trend of computer proliferation, instructors from all disciplines have been taking computer programming classes. After completion of these classes, several instructors have created computer aided instruction courses specially oriented toward their area of specialization. Concerning both the commercial product and the instructor originated computer aided instruction packages, there is a tremendous amount of redundancy in reinventing the underlying structure for each new block of instruction produced. It is this redundancy which is the basis for this study. The published research on this issue is virtually nonexistent from the commercial vendors and the information from instructor originated computer aided instruction is usually undisciplined and inefficiently organized.

Need for the Study

Because of the lack of published algorithm packages concerning computer aided instruction modules, it becomes imperative that computer scientists should publish algorithm packages in this area as tools for the academic community to build on. This algorithm package should serve as a computational recipe to those instructors that want to program their own computer aided instruction modules. This algorithm package should also prove useful for the development of training packages for personnel in both the industrial and military environments.

Objectives of the Study

Objectives are to perform an exploratory study concerning the following:

To design an algorithmic computer aided instruction package which will:

a) serve as a mechanism for the creation of an assignment presentation system by a course instructor

b) serve as a mechanism for the creation of a response evaluator for interactive responses entered by a student

Assumptions for the Study

The assumptions made for this study are as follows:

a) Currently, there are very few commercial products available which perform a function similar to the designed algorithm package.

b) Most computer aided instruction modules are created by instructors who have learned a programming language secondary to their primary profession, and generally these computer products are oriented toward a specific topic.

c) There are very few clear and understandable published algorithms concerning computer aided instruction packages which serve as templates for programmers and educators to follow. Thereby creating an environment where the "wheel is constantly being reinvented".

d) Instructors are thoroughly knowledgeable with their particular area of specialization to establish a presentation system consisting of text, questions and answers.

CHAPTER II

REVIEW OF LITERATURE

Historical Perspective

Over the last couple of decades, computers in education and industry have experienced an unprecedented growth. The number of computer applications seems boundless, and its usefulness is of an infinite magnitude.

Since its conception, computer aided instruction has experienced an exceptional growth throughout the 1960-70's, thus stimulating the growth of a new (Watson, 1972) education discipline known as the area of education technology. Within this new discipline there were two evolving conceptualizations; first, the education process could be improved through the more efficient use of mechanical tools and instrumentation (i.e., TV, motion pictures, etc.). The second conceptual line of thought was that to improve the learning process, the theoretical framework of communication theory must be associated with the long time desired goal of individualized instruction and structured into a format of specific objectives accomplished by learning machines. Another influential factor promoting computer aided instruction was the rapid growth of the data processing industry. There was hardly any facet of the surrounding world (Robinson, 1970) which has not been affected by this industry. Pioneering efforts into the possibilities of instructional computing (Atkinson and Wilson, 1968) was conducted at the University of Illinois on the PLATO I educational system. With the development of time-sharing computer systems, the research of the PLATO I system was enhanced by allowing instructional presentations to be available to many users at once. The federal government established its dedication to instructional computing through the passage of the Elementary and Secondary Education Act of 1965 which provided monetary resources for further computer aided instruction research.

Commercial vendors (Watson, 1972) such as IBM, RCA, and Philco-Ford, contributed immensely to this new instructional presentation concept. Several grants initiated exploration into this arena at Stanford University, Florida State University, and Dartmouth (Chambers and Sprecher, 1968).

Need For Computer Aided Instruction

A belief as to why computer aided instruction is such an interesting topic in modern society goes back in time to one of the basic goals of the education process, that is the achievement of providing individualized instruction. Through the use of computer aided instruction this goal is being obtained, however there is sufficient room for improvement. Course presentations have been created to support class lectures and assignments, and also to stand alone as a totally separate block of instruction.

A supporting factor which contributes to the need of computer aided instruction is the growth of information availability. The computer can purge great amounts of information in a relatively short period of time thus providing easy access to desired information. The computer also allows itself to be utilized as a tool (Willis, Johnson and Dixon, 1983; Taylor, 1980). It performs many manual tasks more efficiently and expeditiously than the old counterparts could perform the function, for example a computer can perform many mathematical calculations in a reasonably short period of time as compared to slide rule or pencil and paper techniques. This efficiency and ease of use produced by this tool allows an instructor to provide more in depth information concerning a topic, than how to do the mechanics of the topic.

Another requirement which makes computer aided instruction desirable is the shortage of qualified teachers (Hickey, 1968). Computer aided instruction allows instructional presentations to fill in the gaps created by inundated faculty and the ability to simulate conditions difficult or impossible to reproduce in the education environment.

Disadvantages

Major disadvantage of computer aided instruction presentations are the lack of structure and the ability to create a robust block of instruction. This particular situation has done more to mar the image of computer aided instruction than any other possibility. A particular failure of the authors designing computer aided instruction is not targeting the audience, therefore leading to the creation of oversimplified block of instructions, which produce immediate boredom and apathy toward this type of product. Another adverse condition is the high initial cost of implementing computer aided instruction. Specific machinery may be required and may only run particular language implementations. Additionally, the actual production of computer aided instruction screen presentations is extremely expensive. An estimate of "over 100" hours of author time to prepare materials for one hour of student time (Rushby, 1979) indicates a major hurdle of existing computer aided instruction presentations.

Modes of Computer Aided Instruction

There are many modes of computer aided instruction in existence, however they can be segregated into five modes of presentations. The first four modes are universally accepted, however, the final mode of existence, as a form of instruction is subject to much criticism.

Drill and practice is the most common form of computer aided instruction. A student sits in front of a terminal and practices repetitive tasks. This concept is used primarily as a means to emphasis previously discussed material (Ellis, 1974; Bitter, 1984).

Tutorial mode of presentation allows a computer through the use of a terminal to interact with a student in the form of presentation screens. This technique is widely used and is an area of much needed improvement. Strides to modify presentations have produced tutorial subtypes (Computer-Assisted Instruction Guide, 1968; Bitter, 1984). The old original tutorial is referred to as a linear tutorial. Using that style, every student must go through the same set of presentation screens in the same order. The technique does not utilize the capabilities of the computer and has been the subject of much controversy concerning the educational benefit verses the cost. The next subtype is the intrinsic branching tutorial (Computer-Assisted Instruction Guide, 1968). It allows a more individualized block of instruction by branching to the next presentation screen based off the previous student response. Finally, the adaptive tutorial response is the most advanced technique in that it determines where to branch for the next presentation screen based off a series of previous answers (Computer-Assisted

Instruction Guide, 1968).

Simulation presentations are the most realistic techniques used to present computer aided instruction. This mode allows the illustration of situations which would be difficult, expensive, or dangerous to replicate in the real world (Bitter, 1984; Willis, Johnson, Dixon, 1983).

Problem solving computer aided instruction is the utilization of the computer to perform some type of manual function automatically. An everyday example of this process is the performance of a mathematical equation. The computer is simply used as a tool to facilitate the learning endeavor (Taylor, 1980; Bitter, 1984).

The previously mentioned controversial mode of computer aided instruction is the game(s) concept. Some educators associate games as an extension of the aforementioned simulation mode, while others do not recognize its existence in education.

Computer Aided Instruction Production

Currently, there are three ways of producing computer aided instruction systems. That is to utilize an existing authoring system, to utilize a special authoring language and/or a general computer programming language (Burke, 1982).

The computer aided instruction authoring system is the most advanced technique for producing automated instruction, however this technique is still in its infancy. The authoring system can be visualized as a package of software programs, written in any language. By implying it is at the infancy stage, only means this technique is still regimented to doing only particular actions in a certain presentation mode. This system is very wasteful of computer storage space, difficult to modify and may not maintain all the data desired. However, it allows an instructor to produce a computer aided instruction product without being knowledgeable of computer science principles.

The computer aided instruction authoring language option allows an instructor more flexibility than the authoring system. However, it requires a larger amount of time to become proficient in utilizing this technique. This method of instruction is similar to using a regular computer programming language except that it possesses special techniques which allows certain aspects of computer aided instruction production to be more efficient (Burke, 1982; Tagg, 1981).

The final approach to developing computer aided instruction is through the use of a regular computer programming language. This method allows the maximum flexibility to the instructor, but requires the greatest amount of time to produce. The technique requires an extensive amount of computer literacy covering facets of the computer language, data structures and storage techniques. It is this approach to computer aided instruction development that this study is concerned.

Instructor Authoring

An important issue in the development of computer aided instruction is the determination of what objectives are to be accomplished by its development. Several situations (Dimas, 1978) which must be determined in advance of programming are, should the computer aided instruction be self sufficient, or should it provide subject reinforcement (Spitler and Corgan, 1979), should the computer aided instruction provide course enrichment and/or some type of remedial learning. Once the goal has been established, then it must be implemented into a structure which facilitates the learning process. Spitler and Corgan (1979) consider eight areas essential in authoring good computer aided instruction programs. First, an index is required to perform the function of a table of contents. This index should provide information as to how long a subject will take to complete and provide the student access to a particular block of instruction if desired by the student. Second, a fine index should break down a subject into topics and provide access to them directly. This allows a student limited in time to pick an area to study. Third, the concept of a pretest at the beginning of each subject in order to determine the students competency in the subject matter. Fourth, the presentation of lesson objectives so the student can have a conceivable idea of what is to be learned. Fifth, is the actual lesson. Sixth, an assisted quiz which utilizes the facilities of the computer reinforcing material previously covered. Should a student answer with a wrong answer, then the diagnostic ability of the computer analyzes the response and provide feedback in helping the student to understand the principle being evaluated. Should the student continue to inaccurately respond, then hints and the correct answer with an explanation should be given. Seventh, a lesson summary should be provided to insure the student achieved what was expected. Finally, the eighth essential element is the presentation of a final exam covering the aforementioned subject matter.

Programming Structure

Various computer aided instruction articles along with the authors Spitler and Corgan (1979) have summarized several points which make or break the design of computer aided instruction programs. They are:

Instructions must be present throughout the entire course informing students what they can and can not do.

Course feedback, one of the leading weaknesses in computer aided instruction presentations. If the answer is wrong, why is it wrong, and what is the correct answer. If the answer is correct, why is it correct.

Branching, an essential element of modern day computer aided instruction. It is the essential element which individualizes a course of instruction based upon the demonstrated knowledge of the student.

Availability to mail messages to the instructor informing of text and program errors. This also allows feedback to the instructor concerning students appraisal of the course.

Ability of student participation within the computer aided instruction program. Provide mechanisms which allow the student to control the flow of the instruction by paging forward, backwards, or going to specific locations.

Provide a large question and/or answer bank. Several questions may have a single answer and vice-a-versa. This allows information to be randomly generated, thus reducing redundancy in the eyes of the student.

The computer aided instruction program should be easy to operate by the targeted audience. Requirements specifying prior knowledge of computer operations should not be required.

Students should be monitored as a means of determining the strengths and weaknesses of the computer aided instruction program and to provide results of their performance.

Programming logic of the computer aided instruction presentation must be reliable and resilience (Rushby, 1979). Computer terminals and programs must be capable of tolerating mishandling without faltering, and if they do they must be repairable quickly.

CHAPTER III

COMPUTER AIDED INSTRUCTION ALGORITHM PACKAGE

Introduction

The study was concerned with the design of a computer aided instruction algorithm package which was intended to serve as a fundamental building block in the development of interactive course presentation systems. This algorithm package is comprised of two unique algorithm modules. One module will facilitate the creation of a course presentation system, and the other an interactive response evaluation system.

Explanation of Terminology

Throughout this study, terminology will be used which may be unfamiliar or confusing in its applied definition. Therefore, to eliminate any discrepancies associated with specific terms utilized, definitions are provided to accurately portray the meanings of these terms.

The term "screen" or display screen (synonymous with "frame" in some literature) represents a snapshot picture in time of an area of space equivalent to the size of a computer terminal's display screen. The size of a screen can vary greatly depending upon the computer terminals manufacturer and its intended purpose. Some terminal screens are 60 characters wide by 23 lines in length compared to others which are 132 characters wide by 25 lines deep. Regardless of the size, a screen can contain any wording, shape, coloring or image. Figure 1 graphically illustrates the concept of a display screen.

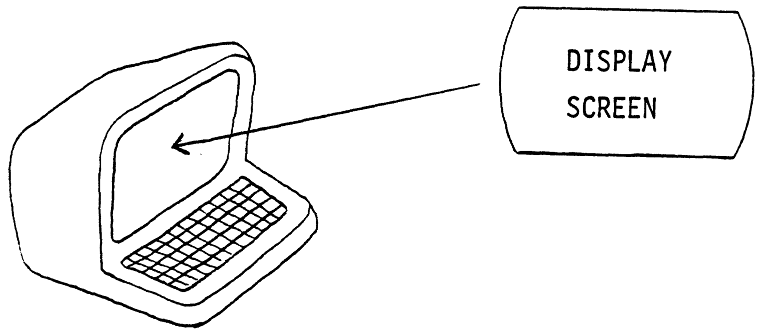


Figure 1. Illustration of a Display Screen

A "subject" is defined as a major block of instruction as defined by the instructor. Several subjects may make up a particular course. In addition, a subject may consist of several subcomponents referred to as "topics".

"Topics" are described as the lowest unit of instruction on which knowledge can be portrayed in this algorithm package. Figure 2 represents an example of subjects and topics.

COURSE: How to Construct a Building

- SUBJECT 1: Site Preparation
    - TOPIC 1: Clear land of brush and trees
    - TOPIC 2: Escavation required
- SUBJECT 2: Construct Building
    - TOPIC 1: Foundation construction
    - TOPIC 2: Wall construction
- etc.

Figure 2. Illustration of Subjects and Topics

It should be noted the definitions given for subjects and topics are somewhat abstract in their meanings to allow the maximum flexibility in adapting to an instructor's course development desires. However, for this study, these terms should be considered as guides to provide a means of laying out instructions in a regulated and formated fashion.

Orientation of Design Concept

Before an implementor can develop this algorithm there must be an understanding of how an instructor will organize a course of instruction. Initially an instructor must create the presentation system consisting of text screens with questions and possible text answers for each subject and topic to be taught. Once these have been established, a determination must be made of what instructional display screen will be next. Details concerning how to format and construct display screens, both artistically and educationally are beyond the scope of this study since these are areas worthy of their own research.

Now, assume an instructor has developed his course adequately and is ready to utilize the assignment presentation algorithm to create the interactive course. Before discussing what the instructor must do next, visualize the underlying structure of this algorithm package and look at the major conceptual structures which serve as its foundation. The first structure can be thought of as a matrix configuration consisting of a grid system composed of the previously defined subjects and topics. The other structure is represented as an on-going list of possible answer responses maintained in some type of empirical order (Figure 3).

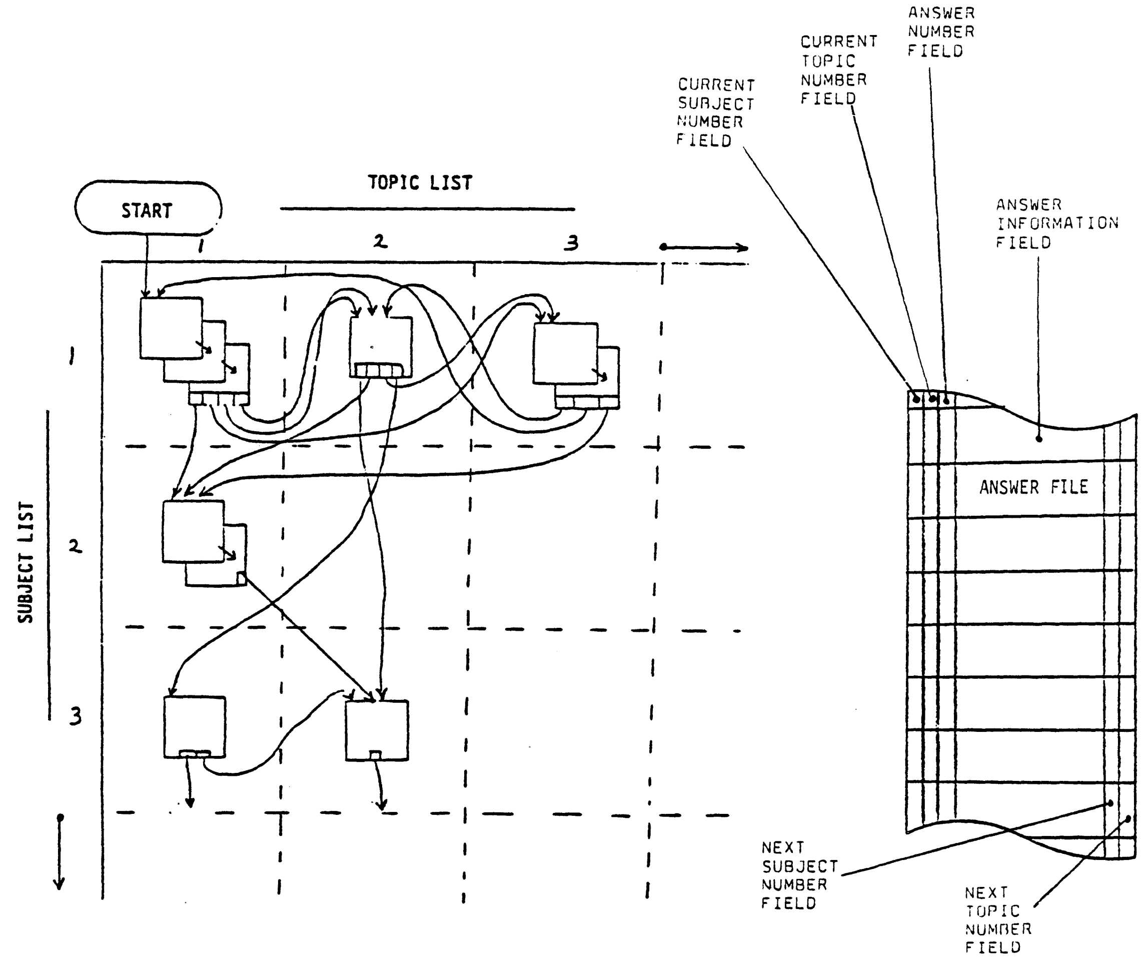


Figure 3. Conceptual Design Structures of this Computer Aided Instruction Algorithm

Notice one of the underlying structures is displayed as being representative of a matrix configuration of size m X

n. m, n are symbolic representations concerning the size of the matrix.

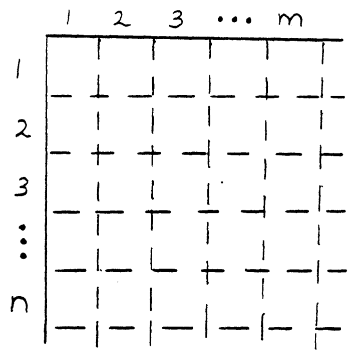


Figure 4. Representation of an m X n matrix

For this design topology, each row represents the presentations required to cover a particular subject of a course. Whereas each column represents different topics of the concerned subject. A different way to think of subjects and topics is to assume a subject consists of five components (i.e. A, B, C, D, E). These five components can then be thought of as being synonymous with the five topics. Look at Figure 5 as a representation of a subject area consisting of five topics. Do not be concerned with the arrows annotated between the topics at this time.

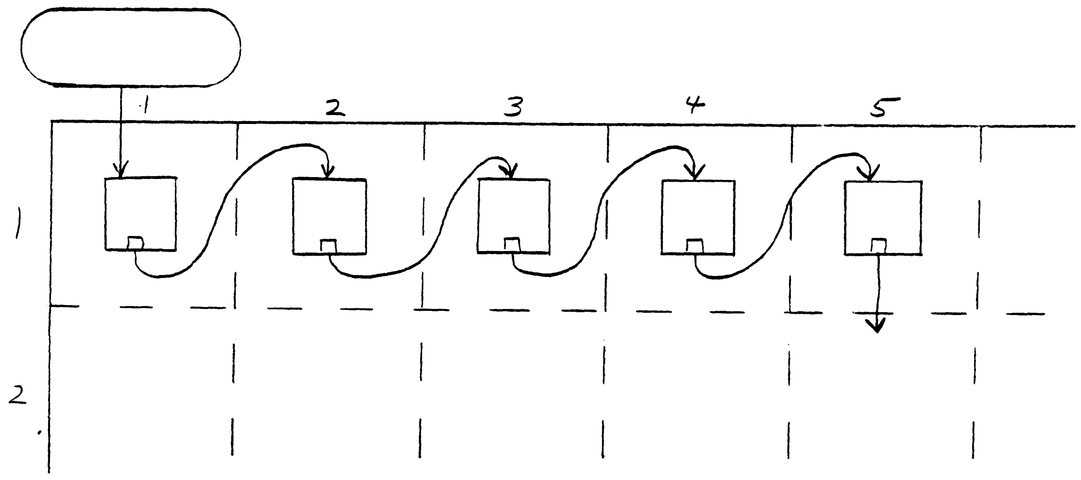


Figure 5. Representation of a Subject Consisting of Five Topics

Next, realize that each grid position of the matrix (for an example, the position represented by the intersection of row 1 and column 1 in Figure 3) can represent a particular set of display screens of a subject for covering a topic. In short, this indicates that a topic cannot be fully discussed on one display screen and that additional display screens must be utilized. Each display screen of information is represented in this algorithm by a box symbol. Refer to Figure 6 which illustrates one grid position on the subject/topic matrix consisting of more than the one topic display screen of information.

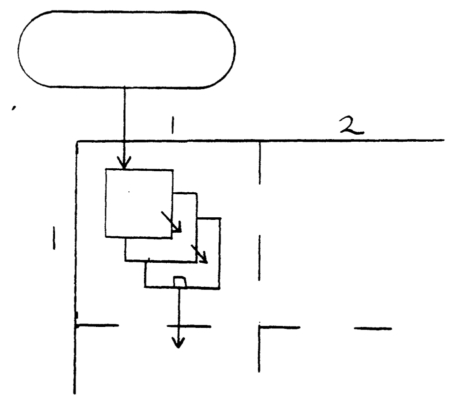


Figure 6. Topic Presentation Consisting of Three Display Screens

Notice within the last screen of each topic position of the grid matrix there is at least one smaller box. This notation is to exhibit the number of possible paths that exist from this topic to the next topic to be instructed. The arrows indicate the possible logical flows of control from one display screen to the next display screen. These paths can be associated with a question being asked on the most recent topic screen, and the mapping to an answer on the aforementioned answer structure. Which in turn specifies the next topic of instruction (Figure 7). Another association given to these paths is when the most recent screen notation only contains one small box, this represents the instructor's desire to branch directly to another topic screen without having to access the answer list structure. Refer back to Figure 3.

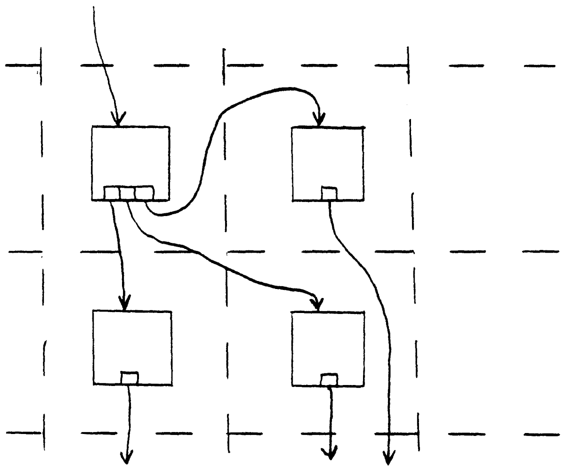


Figure 7. Display Screen With 3 Possible Answers in Which to Branch to Next Topic Display Screen

Algorithm Tools

In order to develop this algorithm, several tools are required to allow the clear and concise presentation of this algorithm's flow. The chosen method to represent this study's algorithm is through the use of flowcharting symbols which conform to the International Organization for Standardization Recommendation R1028 Flowchart Symbols for Information Processing. The choice to illustrate this algorithm by using flowchart symbols is based upon this author's belief that they are easily understood by most persons capable of relating to logical structured thought patterns. The flowchart representations utilized will consist of a set of symbols which will indicate particular operations to be performed (Figure 8).

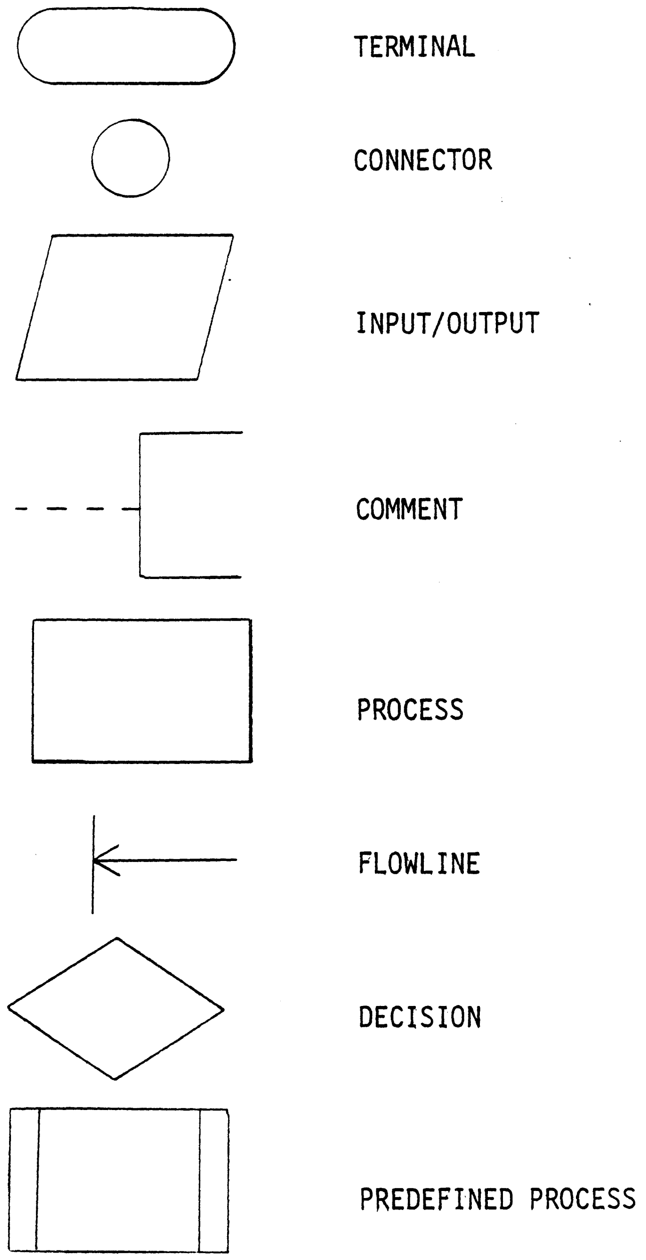


Figure 8. Flowchart Symbols

a. The terminal symbol is used to indicate a beginning or end point in an algorithm, or a return from a subprogram.

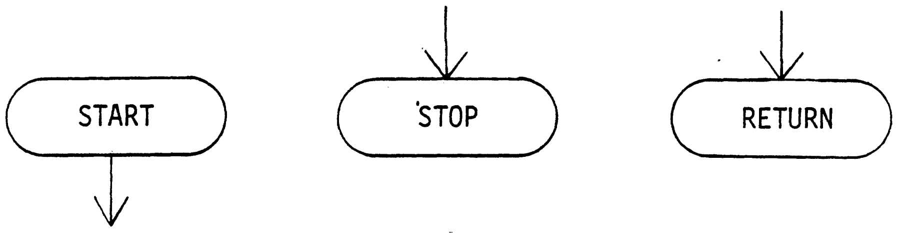


Figure 9. Terminal Symbol

b. The connector symbol is used to signify when the logical completion point has been reached, and for an exit from one page to an entry point on another page. Connector symbols are connected to the line of flow symbols. Labels placed within connectors serve as a means of identifying algorithm control flow.

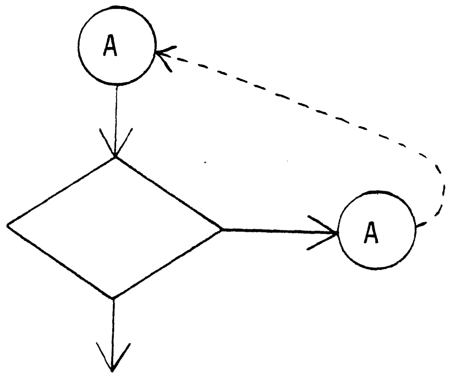


Figure 10. Connector Symbol

c. The input/output symbol indicates some type of input/output operation which allows information to become available to the algorithm or information produced by some action of the algorithm.

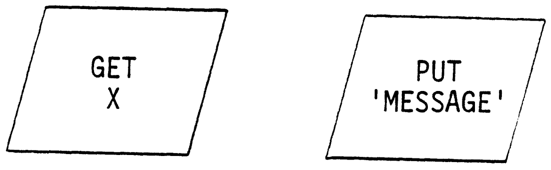


Figure 11. Input/Output Symbol

d. The comment symbol allows additional descriptive information for clarification purposes. It is attached through the use of a dotted line to any symbol or flowline as required.

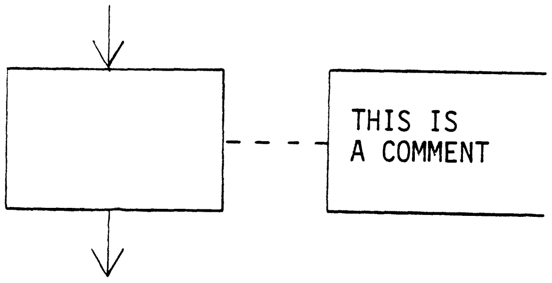


Figure 12. Comment Symbol

e. The process symbol reflects any operation performed which changes values, locations, or assignments of information. It can contain any number of these manipulations.

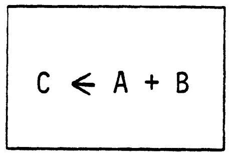


Figure 13. Process Symbol

f. The flowline symbols reflect the sequence of operations from beginning to end. Normal procedural flow is from left to right and top to bottom. For clarification, arrow heads may be used to reduce any possible confusion.

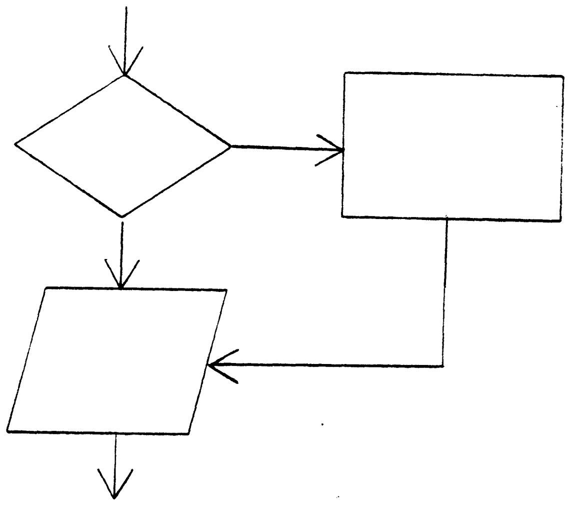


Figure 14. Flowline Symbol

g. The decision symbol demonstrates the changing of flow of control or switching operations to other alternative choices which proceed with the operation.

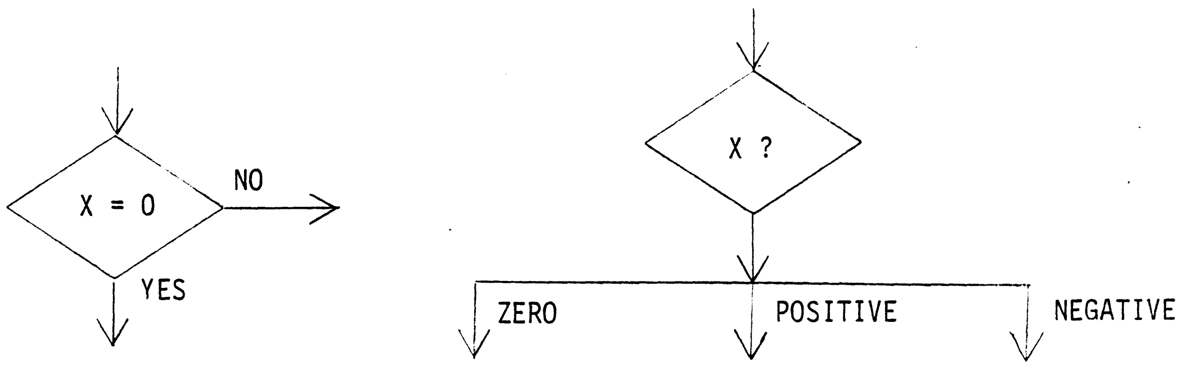


Figure 15. Decision Symbol

h. The predefined process identifies one or more subprograms specified on another flowchart.

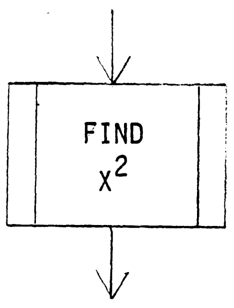


Figure 16. Predefined Process Symbol

Besides the above flowcharting symbols which are fairly standardized, there are several unique forms of notation which must be understood for this study.

The "assignment operator" signified by a (<-), illustrates the movement of data from one position or form to another position or form.

Example	Take the value designated as Y and place that value in the place of X, overwriting
$X \leftarrow Y$	whatever was previously in X.

Figure 17. Example of Assignment Symbol

Take the value designated as Y and place that value in the place of X, overwriting whatever was previously in X.

The "comparison operator" is signified by a colon (:) which represents a binary operation and usually involves the comparison of two quantities. This operator is generally found within decision blocks in order to alter algorithm flow of control.

```
Example
   X : 0        Compare X to zero.
```

Figure 18. Example of Comparison Operator

The "concatenation operator" is represented by a (&) sign. This operator will join two entities together forming a single entity. Applications of this operator usually applies to string values.

```
Example
   AB ← A & B
```

Figure 19. Example of Concatenation Operator

The "unconcatenate operator" is illustrated by a vertical bar (|) which causes a single string entity to be separated into two or more entities.

Example

$$A, B \leftarrow A \mid B$$

Figure 20. Example of Unconcatenate Operator

Summary

This chapter highlighted the terminology and logical representations which take place in the remainder of this study. A critical point pertinent to this algorithm's study is the conceptual understanding of the basic underlying structures. The grid matrix referenced by subject and topic is a precise way to portray a display screen's exact location in a course and it's relationship with its contemporaries. The answer list structure allows an infinite number of possible answers to a topic's question(s) and allows an infinite number of mappings from an answer to another topic presentation.

CHAPTER IV

ASSIGNMENT PRESENTATION ALGORITHM

Introduction

The purpose of this chapter is to describe the algorithm design of a computer aided instruction module which will facilitate the creation of an interactive assignment presentation instruction system, relate the overall design concept, and discuss the rational of how the algorithm functions.

General Design Concept

Before being able to produce a computer aided instruction design module as an assignment presentation system, it is important to first understand the general scheme of what the overall architecture of this endeavor looks like. Figure 3 pictorially represents the concept concerning this study's algorithmic approach to a computer aided instruction assignment presentation system. It looks rather complicated. However, it is simply a collection of display screen presentations with a mapping to the next display screen based upon a particular answer or default action. A break-down of what Figure 3 is portraying will reveal the underlying simplicity of this design concept.

Algorithm Development

Algorithm development is concerned with designing a computational recipe in which someone experienced with computer programming could follow to produce some desired result in a finite number of steps. Another perspective of an algorithm is to consider it similar to a kitchen recipe for making cookies. Most cookie recipes call for flour, sugar, nuts, etc.; however, it is up to the cook using the recipe to determine how much of each ingredient to use, if any at all. It is this same idea that serves as the fundamental concept of an algorithm.

In producing the assignment presentation algorithm, we must first consider what actions an instructor would need to create, maintain, and/or remove display screens of information, questions and answers. To begin with a course consisting of no (0) display screens, an instructor would need the capability to create screens of text with questions and answers. Once several screens of information and their associated answers are created, an instructor might want or need to find a certain screen or answer to modify it. Also, an instructor may need to totally delete a currently existing screen or add a new screen of information. These operations would have to be performed without seriously affecting the overall intent of the assignment presentation algorithm. (Refer to Figure 21 for a listing of required functions necessary to produce a computer aided instruction assignment presentation system.)

A. Create a screen w/question(s) or answer(s).
B. Locate a particular screen for some desired action.
C. Edit capability of existing material.
D. Delete ability to remove entire inappropriate screens or answers.
E. Addition ability to insert entire screens or answers within an existing file.

Figure 21. Minimum Essential Functions Necessary In order to Produce Assignment Presentation Algorithm

For the purposes of studying this algorithm you can visualize the physical storage of this information to be similar to Figure 22. In reality however, the physical storage of data manipulated by the implementation of this assignment presentation algorithm could be of any form dependent upon a particular computer's organization. This algorithm is applicable for implementation on real-time mass storage systems to small microcomputers utilizing tape or diskette storage devices. The only critical feature which must be considered during implementation is a determination if sufficient memory exists for the creation of a new screen's information prior to it being created or added to the existing system.

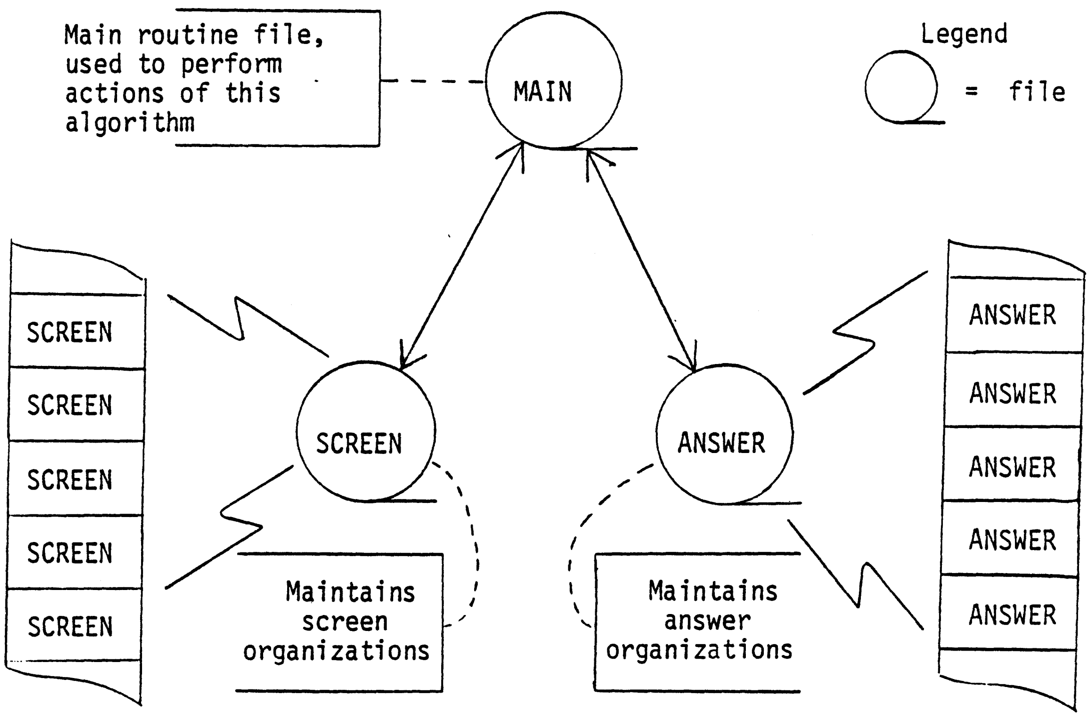

Main routine file, used to perform actions of this algorithm
MAIN
Legend
= file
SCREEN
SCREEN
SCREEN
SCREEN
SCREEN
SCREEN
Maintains screen organizations
ANSWER
Maintains answer organizations
ANSWER
ANSWER
ANSWER
ANSWER
ANSWER

Figure 22. A Way in Which to Visualize Physical Storage of Information Utilized by the Assignment Presentation Algorithm

For this study, conceptualize in your mind the screen and answer file structures shown in Figure 3. Knowledge of these structures will facilitate the understanding of the following algorithm.

To begin the assignment presentation algorithm it is assumed that a course exists and it is desired to perform several operations upon that course.

Part 1 of the Assignment Presentation Algorithm

Part 1 of the assignment presentation algorithm (Figure 23) tries to identify exactly what action an instructor wants to accomplish upon this system.

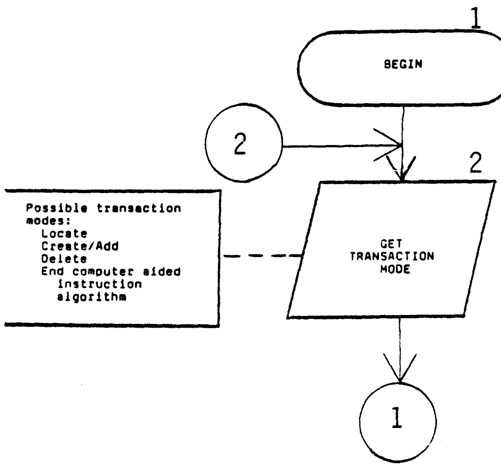

1
BEGIN
2
2
Possible transaction modes:
Locate
Create/Add
Delete
End computer aided instruction algorithm
GET TRANSACTION MODE
1

Figure 23. Part 1 of Assignment Presentation Algorithm

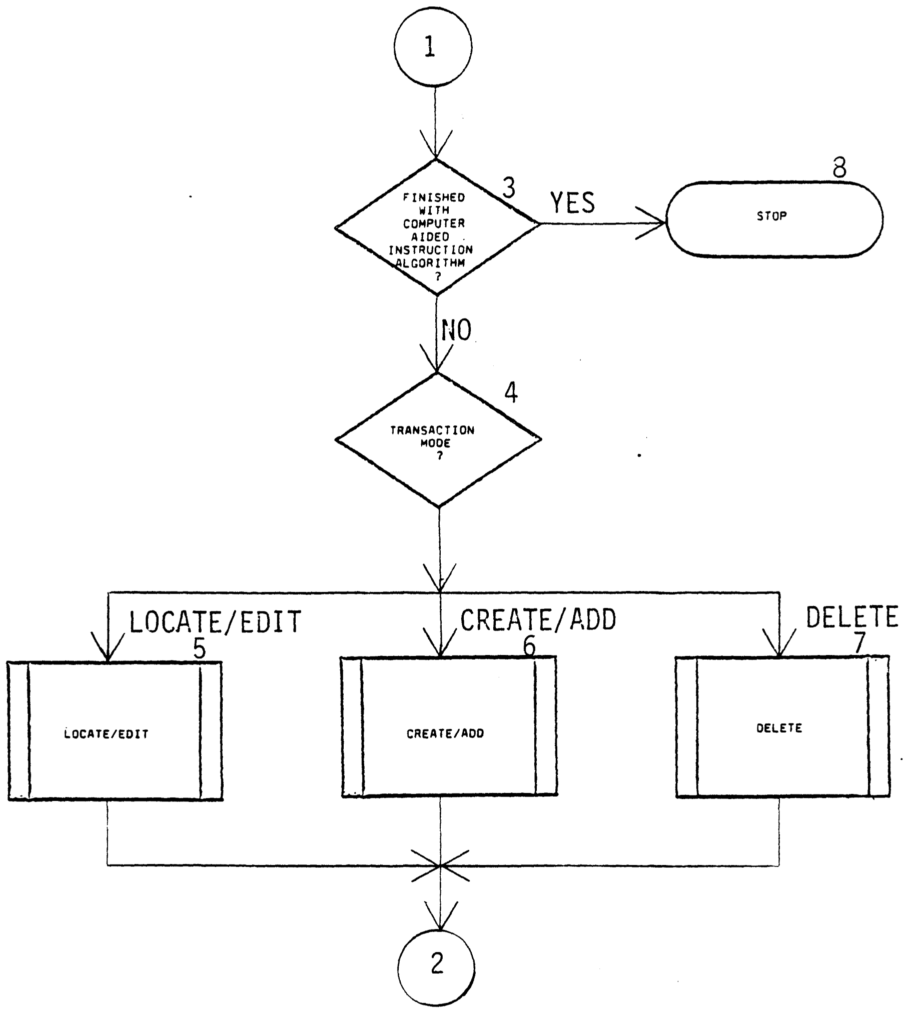


Figure 23. (Continued)

Tracing through Part 1 of this algorithm by the block numbers annotated at the top right of each flowchart symbol will clearly portray what actions are occurring.

Block 1: initiates the assignment presentation algorithm.

Block 2: an input/output block which needs information from the instructor indicating what action is to be performed by the algorithm, if any.

Block 3: determines if the instructor's previous response was to exit this algorithm. If yes, then the algorithm is terminated by proceeding to Block 8. Otherwise, proceed to the next block.

Block 4: determines what action is needed to direct algorithm's flow of control to perform the instructor's desired wishes. Notice the instructor's options have been reduced to three choices, i.e. locate/edit, create/add or delete.

The reduction of choices available to an instructor in this algorithm reflects this author's opinion that combining of the functions, locate and edit, will only improve the readability of the algorithm. For an edit function to be performed, one must first be able to locate the desired screen or answer. If the desired screen can be located, then it is left up to the algorithm implementor to create edit functions dependent upon the application computer system, operating system and/or the instructor's desire. The combination of the functions, create and add, is merely a play on semantics. The process of creating something is exactly the same process as adding something that was previously nonexistent. For the algorithm these two terms can be considered as synonyms of each other.

Blocks 5, 6, 7: are algorithm subprogram calls which will be discussed as Parts 2, 3 and 4 of this algorithm.

Proceeding from this point, the algorithm continues in a loop back to Block 2 until the instructor terminates the algorithm.

Block 8: terminates the assignment presentation system algorithm.

Before introducing the components of this algorithm, refer to the legend at Table I. This table explains all symbols used in the study.

TABLE I

SYMBOL LEGEND OF COMPUTER AIDED INSTRUCTION ALGORITHM PACKAGE

A	Variable for answer file subject identification.
AF	Answer file.
AFN	Answer number in answer file.
AIF	Answer information field in answer file.
B	Variable for answer file topic identification.
BMP	Pointer to bottom (end) of subject list.
C	Variable for particular answer in answer file.
CAI	Computer aided instruction terminal screen display.
CSNF	Current subject number field in answer file.
CTNF	Current topic number field in answer file.
D	Variable for answer file identifying next subject identification in which to branch.
DP	Header node down pointer.
E	Variable for answer file identifying next topic identification in which to branch.
FIRST	First node on any list.
GP	Generic pointer--points to next available node on a one way linear list.

TABLE I (Continued)

HEAD	Initial point of origin for subject list.
HN	Header node.
I	Location of next available node.
LIST	Generic list--any one way linear list.
LOC	Current location on subject list.
N	Header node subject number.
NEW	Symbolizes the newly created node.
NHN	Next header node on subject list.
NN	Next node on list.
NSNF	Next subject number field in answer file.
NTN	Next topic node on topic list.
NTNF	Next topic number field in answer file.
P	Pointer to data areas within header node.
PHN	Previous header node on subject list.
PRN	Previous node on list.
PTN	Previous topic node of topic list.
SCN	Screen list.
SI	Computer aided instruction screen information.
SL	Subject list.
SN	Screen number of topic node.
SP	Screen pointer, points to next node in screen list per a subject and topic identification.
T	Pointer to data areas within topic node.
TEM	Temporary location marker.

TABLE I (Continued)

TL	Topic list.
TMP	Pointer to top (beginning) of subject list.
TN	Topic number of node.
TOP	Topic node.
TP	Topic pointer, points to next node in topic list.
UP	Header node up pointer.
X	Variable for course subject identification.
Y	Variable for course topic identification.
Z	Variable for course screen identification.

Part 2 of the Assignment Presentation Algorithm

The first algorithm of Part 2 to be discussed is the Locate/Edit Subprogram algorithm. It is the purpose of the algorithm to search through the various lists by subject, topic and screen identifications and determine the presence of the requested display screen. It is assumed henceforth that these lists are referenced by the string of natural numbers (0, 1, 2,...,).

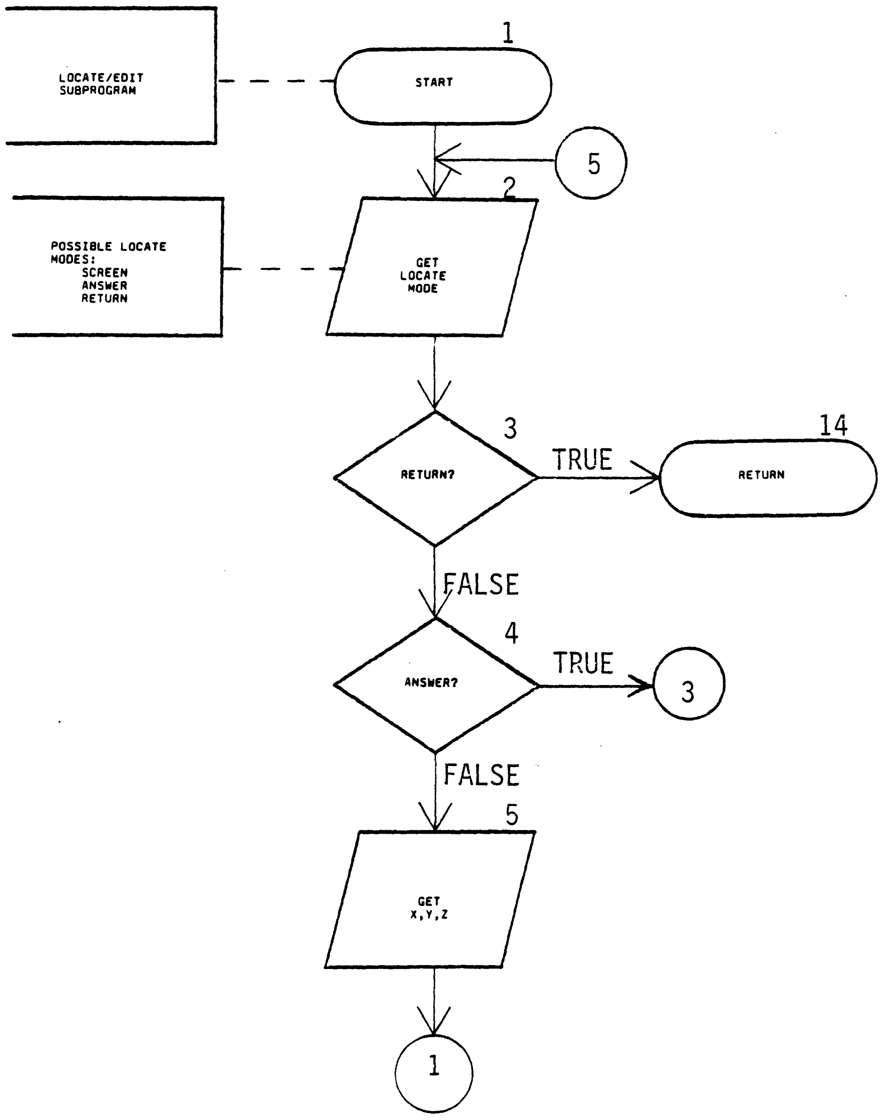


Figure 24. Locate/Edit Algorithm Subprogram

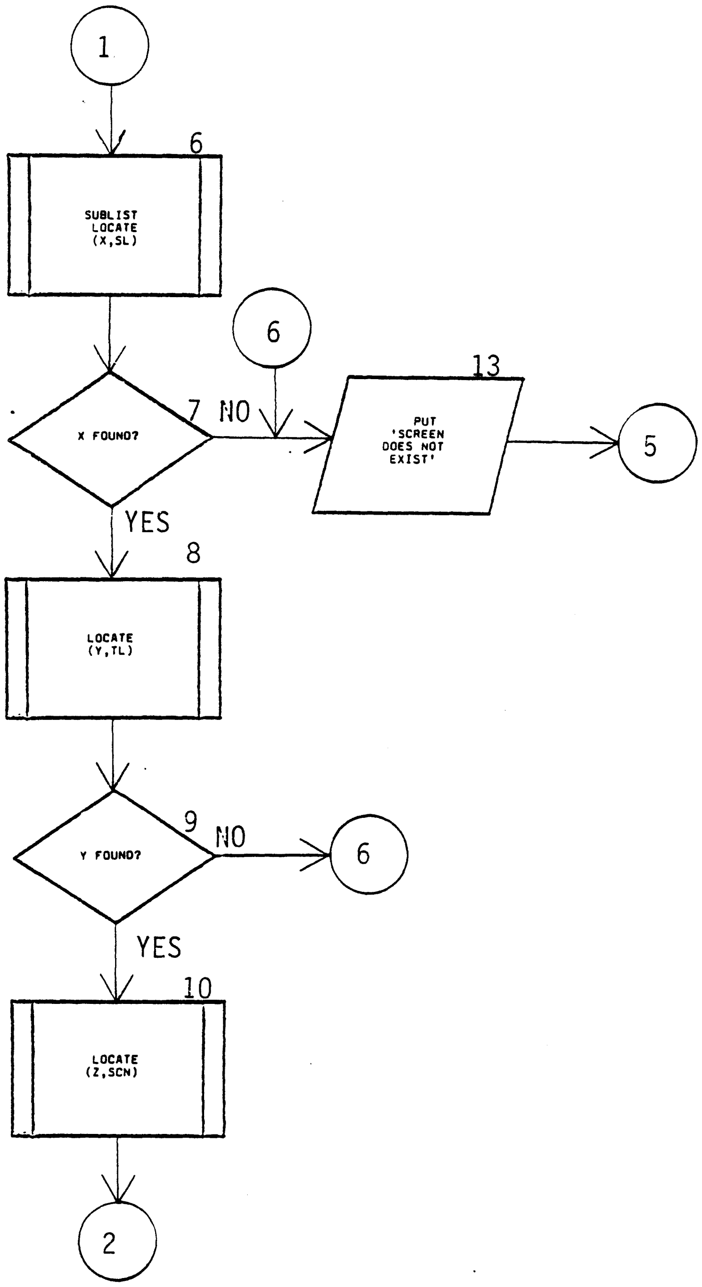


Figure 24. (Continued)

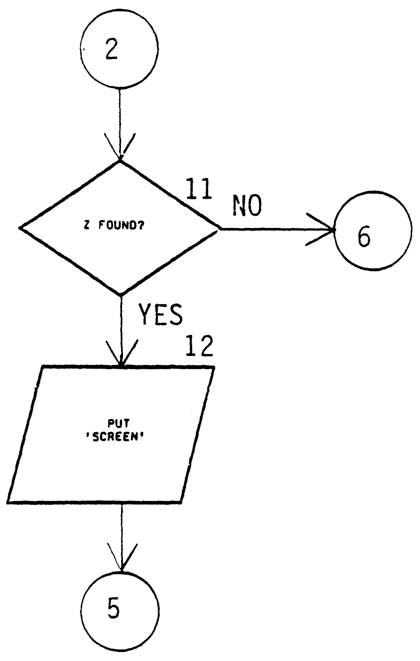


Figure 24. (Continued)

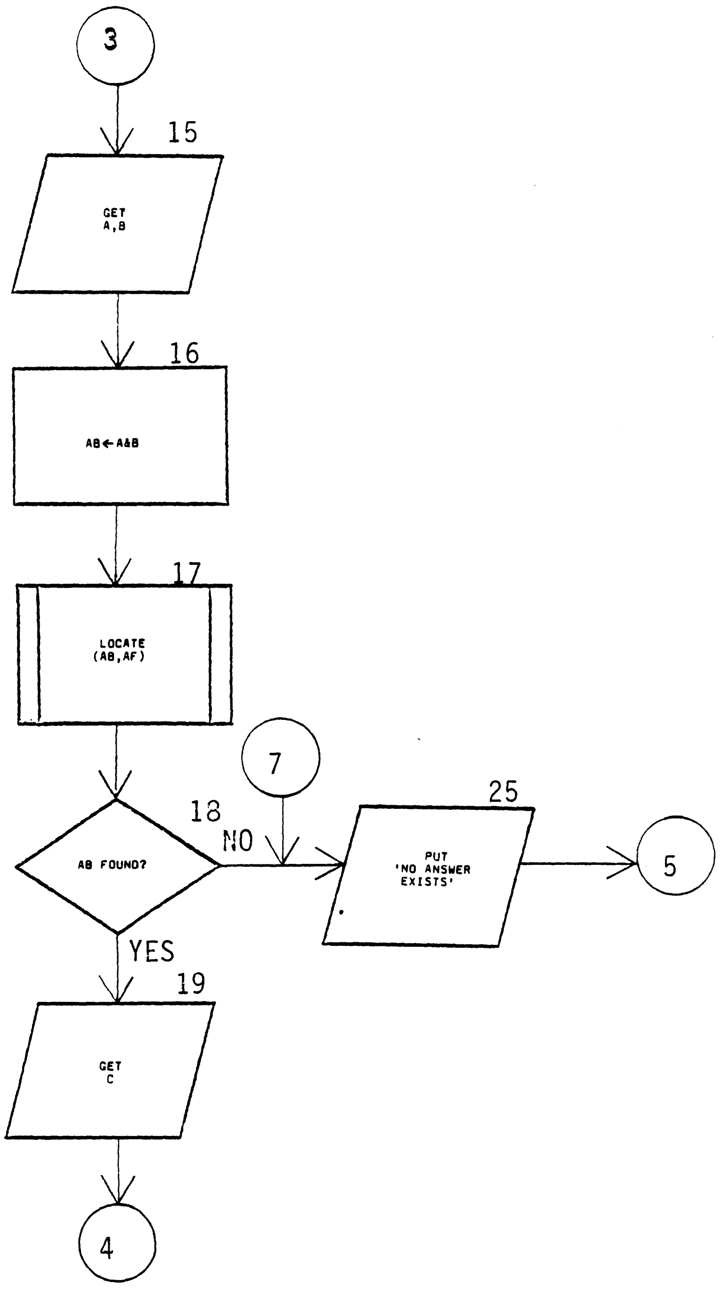


Figure 24. (Continued)

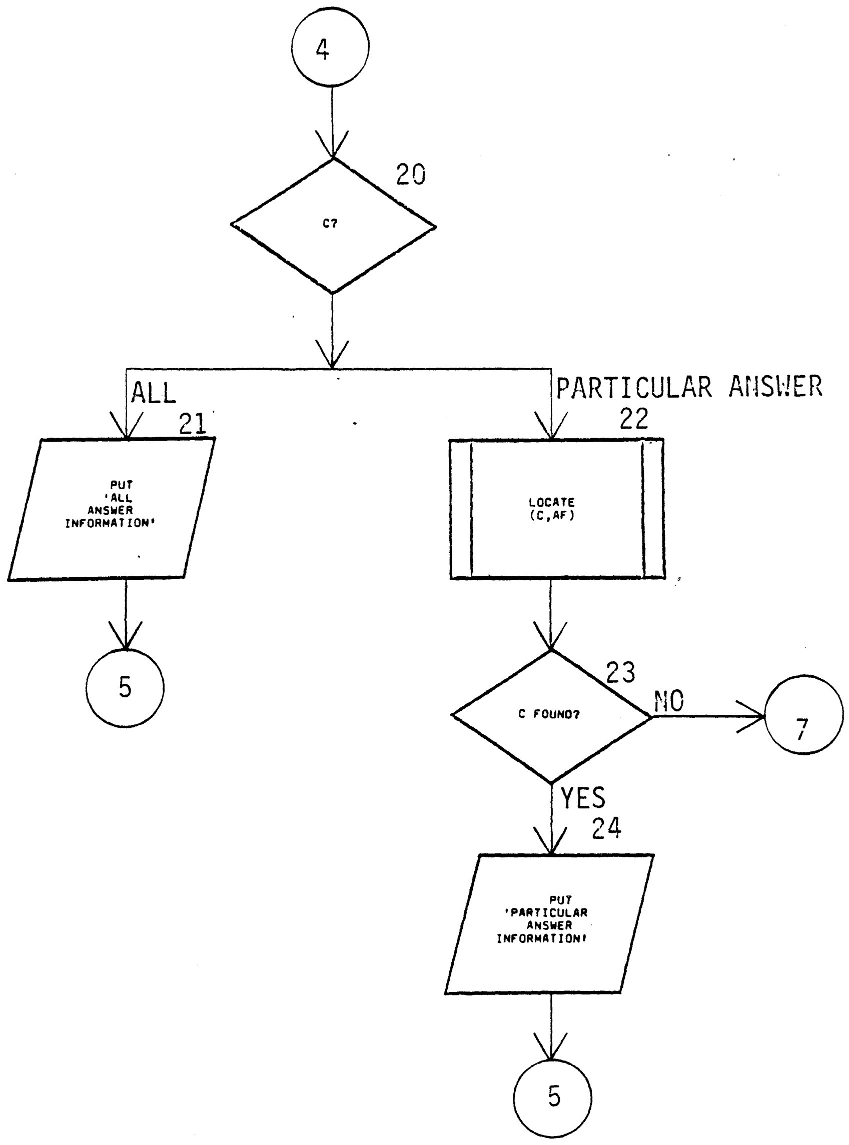

4
20
C?
ALL
21
PUT 'ALL ANSWER INFORMATION'
5
PARTICULAR ANSWER
22
LOCATE (C,AF)
23
C FOUND?
NO
7
YES
24
PUT 'PARTICULAR ANSWER INFORMATION'
5

Figure 24. (Continued)

Block 1: beginning of subprogram algorithm.

Block 2: requests information from the instructor as to whether to locate/edit some data on the screen or answer file, or to exit from the subprogram call.

Block 3: is a decision block trying to determine if the author is finished with this function prior to being returned to the program's main mode.

Block 4: decision block to determine if the answer file is to be accessed.

Block 5: requests input concerning the screen's location by subject, topic and screen identification numbers.

Block 6: is a subprogram call to a routine which determines if the screen can be located on the doubly linked subject list.

Block 7: decision block to determine if X was found on the subject list. If X is not on the list, branch to Block 13.

Block 8: subprogram call to a routine which determines if Y is on the topic list.

Block 9: determines if Y was found on topic list. If not, branch to Block 13.

Block 10: subprogram call to a routine which determines if Z is on the screen list.

Block 11: was Z found on the screen list. If not, branch to Block 13.

Block 12: Display a terminal screen of information to the user. Algorithm then branches back to Block 2 in order to perform another iteration, if so desired.

Block 13: Same as Block 12 above.

Block 14: return specifies no more action required of this algorithm. Algorithm returns to Part 1 previously discussed.

Block 15: input/output symbol signifying the instructor must input the subject number and topic number in order to locate the answer field information within the answer file.

Block 16: indicates the subject number is to be concatenated with the topic number to form a single natural number.

Block 17: a subprogram call to a linked list routine to determine if AB is in the answer file.

Block 18: was AB found in the answer list (file). If not, proceed to Block 25.

Block 19: instructor must signify if all answers per a certain subject and topic number are to be displayed or only a particular answer.

Block 20: was answer number found in the answer file?

Implementors of the algorithm could insert a loop at this point in the program and be able to select more than one answer to look at.

Block 21: Display all answers addressed by a particular subject and topic identification.

Block 22: subprogram call to a one way list routine that determines if the particular answer requested by the instructor is present in the answer file.

Block 23: was the particular answer found?

Block 24: print out the text contained in the answer field of that particular entry in the answer file.

Block 25: put out a message the sought after answer does not exist in the answer file.

The second algorithm (refer to Figure 25) represents a search through a two way list structure by subject number until the correct location is or passed. This subprogram algorithm must be accessable from all transaction modes applicable to the entire assignment presentation system.

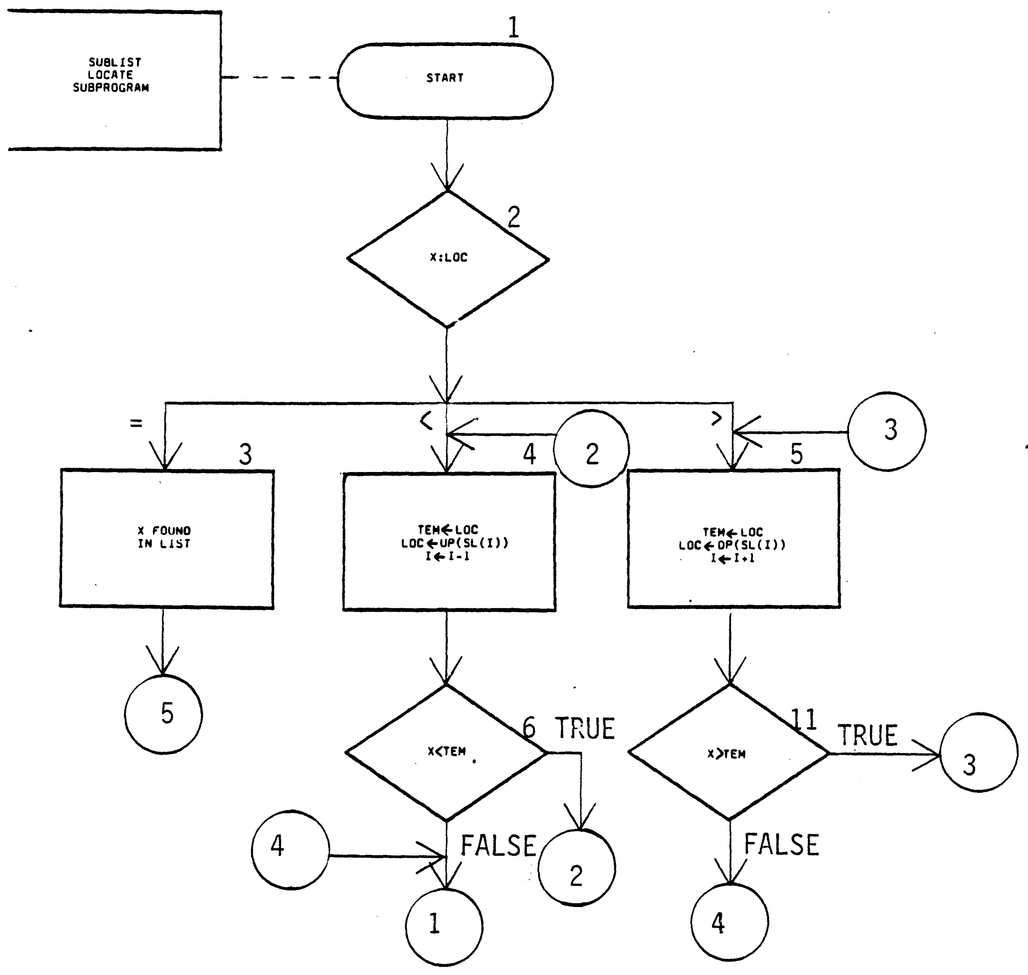


Figure 25. Sublist Locate Subprogram Algorithm

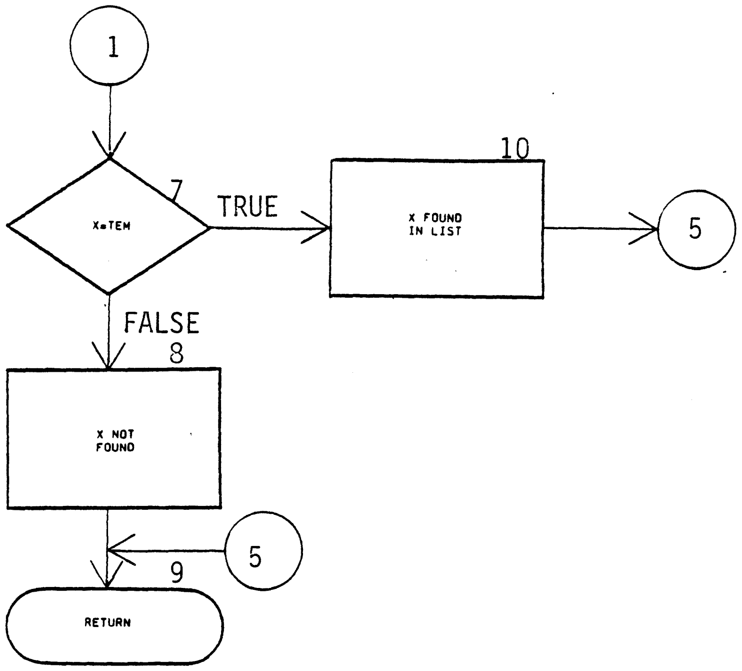


Figure 25. (Continued)

Block 1: begins subprogram called "sublist locate".

Block 2: is a decision block to determine if the desired course subject number is equal to, less than, or greater than the value at current position in the list.

Block 3: signifies the course subject number has been located in the two way list.

Block 4: if the Block 2 decision symbol indicated the course number was less than the current position then the search would have to proceed back up the list toward the beginning, one node at a time. During each step of the movement, the current position is always assigned to a temporary position prior to backing up to the previous node.

Block 5: if the block 2 decision symbol specified a subject number which was greater than the current location marker, the algorithm would require the current location marker to be incremented. First action required is to assign the current position marker to a temporary marker. Then the current position would be incremented down one subject position toward the end.

Block 6: proceeding after Block 4, this block determines if the sought after subject number is less than the current position marker. If so, loop back to Block 4.

Block 7: determines if the subject number sought is equal to the current position marker. If true, then X is found, else X is not found.

Block 8: signifies the searched for value does not exist in the list.

Block 9: leave this algorithm and return to the calling algorithm.

Block 10: means the sought after value has been found.

Block 11: follows Block 5. Determines if the subject number is greater than the current position marker. If true, loop back to Block 5, else branch to Block 7.

The third algorithm of Part 2 (refer to Figure 26) is used to locate a point on either the topic, screen or answer lists. It represents a way in which to traverse a list in a linear order from the current location or point of origination to the desired position or end of list, whichever occurs first.

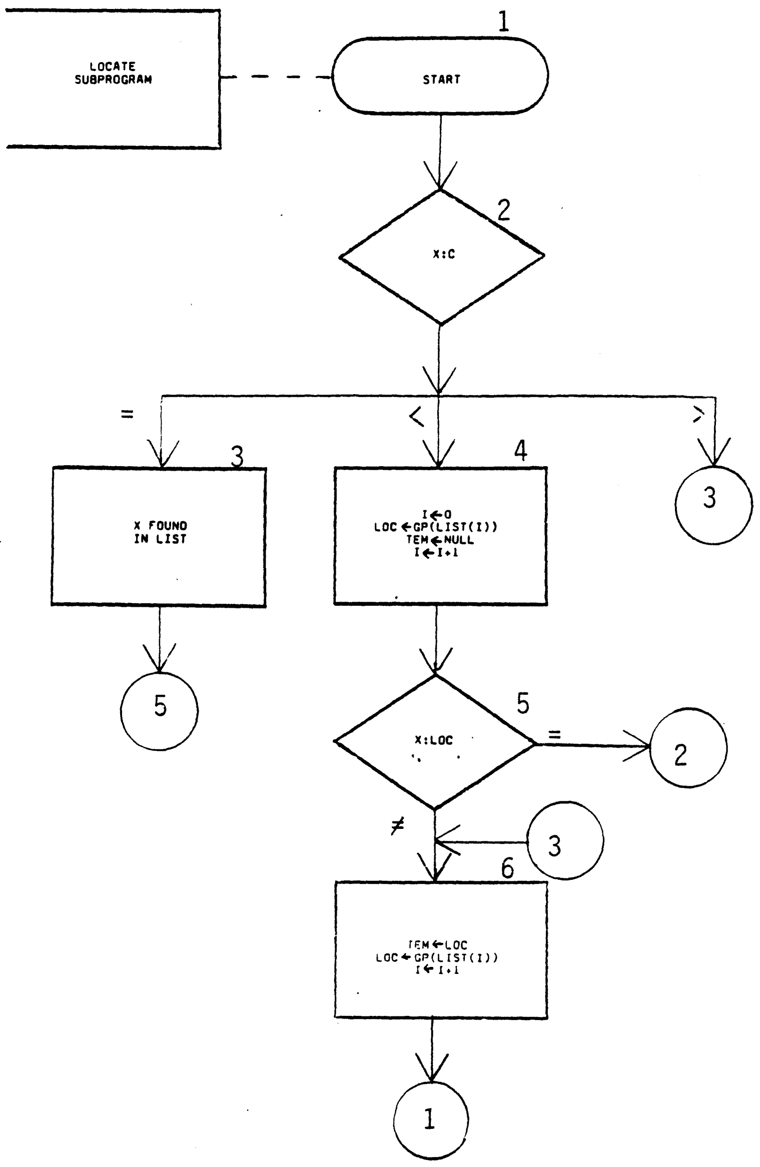


Figure 26. Locate Subprogram Algorithm

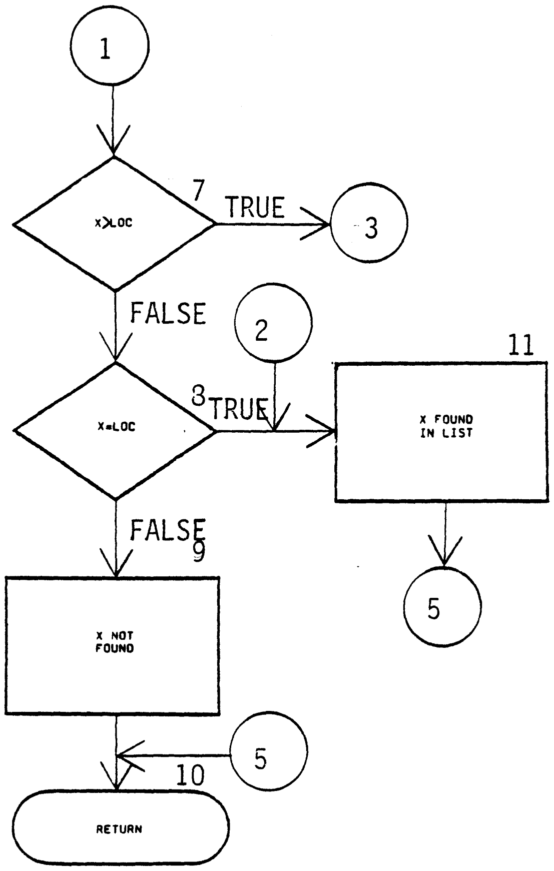


Figure 26. (Continued)

Block 1: begin Locate subprogram.

Block 2: decision block to determine if the sought after number is equal to, less than, or greater than the current position marker.

Block 3: indicates the value sought has been found.

Block 4: if the sought after number is less than the current position on the list then initialize the list back to its beginning and assign the temporary pointer to null. This first position is assigned to the current marker and the pointer is advanced forward to the next node.

Block 5: decision block to determine if the first position is equal to the desired list number.

Block 6: assigns current position to a temporary marker and assigns next nodal information to current position.

Block 7: determines if the desired position is greater than the current position. If so, loop back to Block 6.

Block 8: decision block to determine if the desired position is equal to the current position.

Block 9: searched for value does not exist in the list.

Block 10: depart this algorithm and return to the calling algorithm.

Block 11: searched for value has been located in the list.

Part 3 of the Assignment Presentation Algorithm

Part 3 of the assignment presentation algorithm deals with the creating and adding of new display screens and answers for the computer aided instruction course being constructed. The algorithm should allow for screens and answers to be added at any time during the duration of a course. Figure 27 is a memory aid to emphasize this point prior to discussing Part 3.

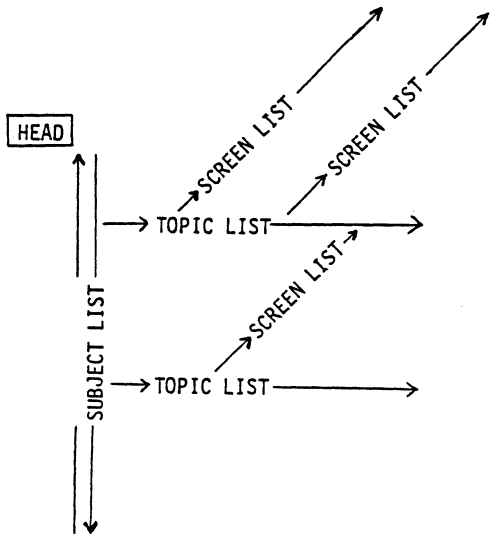


Figure 27. Memory Aid Visualizing List Flow of Assignment Presentation Algorithm

The "Head" equates to the origination point of all subjects on the subject list. For a display screen to exist at least one header node must be on the subject list. This header node represents the subject to be presented and which may consist of numerous topics. Each header node will point to at least one topic node. A topic node is the structure which will actually contain the computer aided instruction screen information in this algorithm. Since a topic may be so large that not all of the information can be contained on one display screen, this requires a screen list. For purposes of this algorithm, the mere existence of a topic node equates to the existence of a screen list of the quantity one. The subject list (refered to as sublist) is portrayed in this algorithm as a two-way list. This means from any position on the list, you may traverse toward the bottom of the list or toward the top of the list. It is assumed in this study the lists are created and maintained in some type of sequential ordering, thereby making decisions easier as to which way to traverse the applicable lists. Both the topic list and screen list can be conceived as representing a linear list structure. Which means the lists are traversed in one direction only. Should a position being searched for lay behind the current position being performed, then the list search would have to start at the beginning of that applicable list. However, I have assumed the instructors authoring computer aided instruction products will more than likely present computer aided instruction display screens in a logically sequential order just as they would as if they were teaching a class. Based upon this assumption, the search time required for searching a list could be minimal since you are probably at the correct location or just a few steps from it forward of your current position.

In order to establish a common base line of understanding concerning the perceptual interpretation of what a header node and a topic node are, refer to Figure 28. This format will be used throughout this entire algorithm.

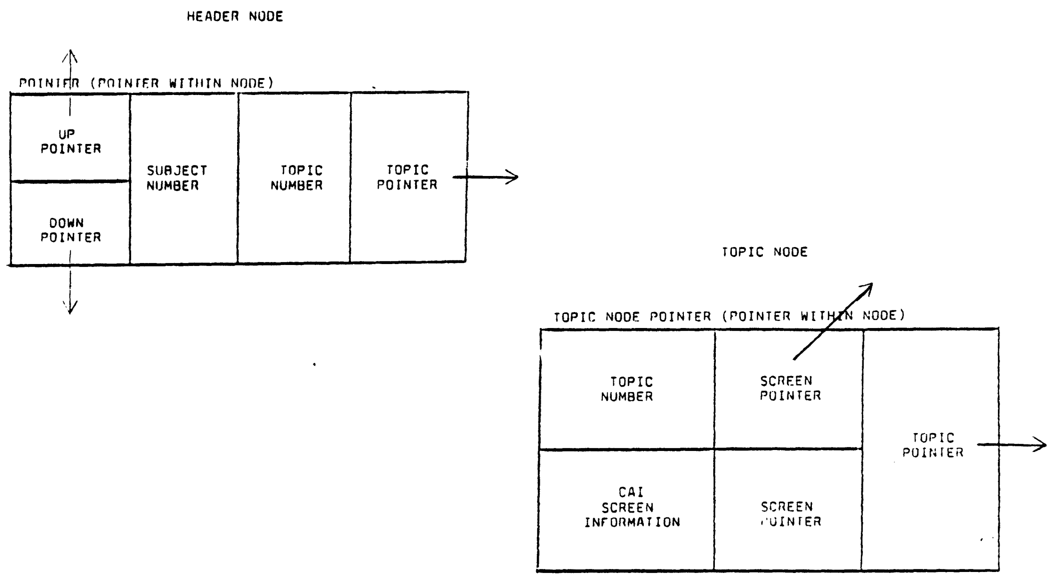


Figure 28. Header and Topic Node Formats

A topic of concern to the implementing instructor of this algorithm is how to physically represent these node structures. This particular challenge is left to the instructor and the capabilities of the utilized programming language. When dealing with certain high level programming languages, they allow a structure called a record. This record structure would allow the implementation of this algorithm to be more easily performed than when compared to a language not possessing this structure. In this case, this situation might require an array of arrays, or any other technique deemed appropriate by the implementor.

When creating or adding information to the answer file there is no absolute need for the information to be in a sequential order, but it could prove advantageous in the economy of time. An alternative applicable to the answer file is to add (and delete) all answers in a random order and when exiting the assignment presentation system to perform some sort routine to realign the sequential ordering. For the purposes of this algorithm, theorize the structure of the record field information to look like that displayed in Figure 3.

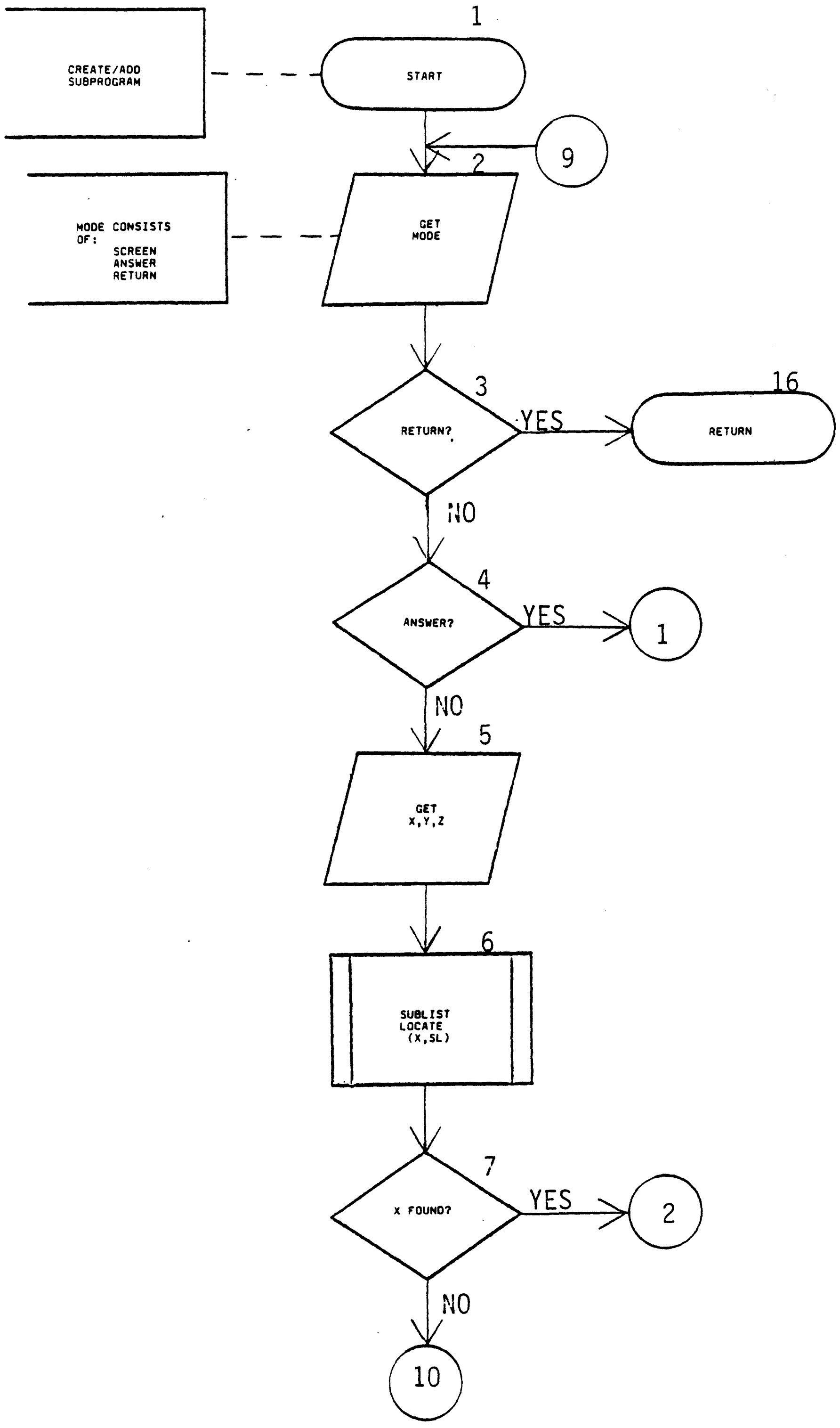


Figure 29. Create/Add Subprogram Algorithm

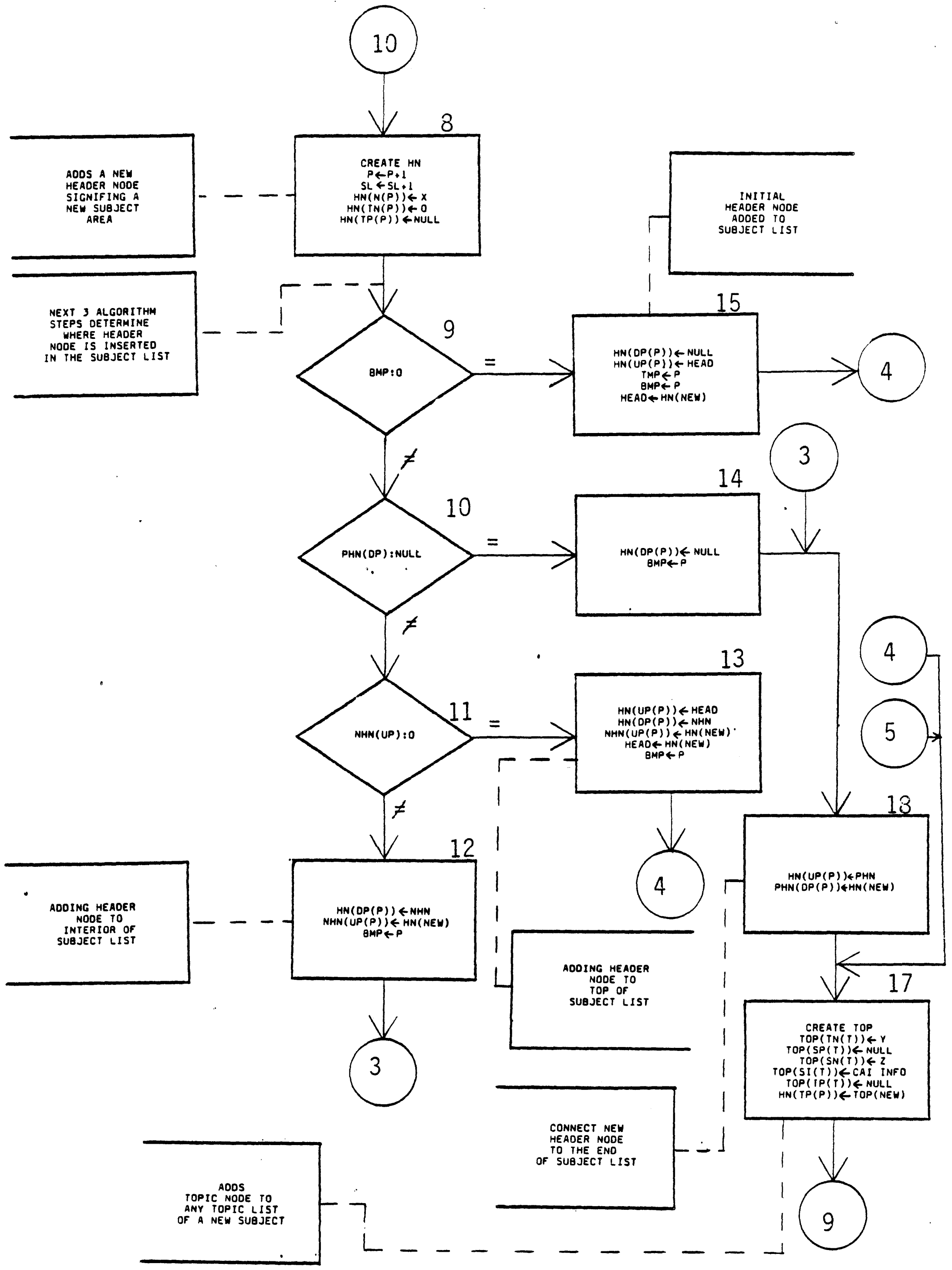


Figure 29. (Continued)

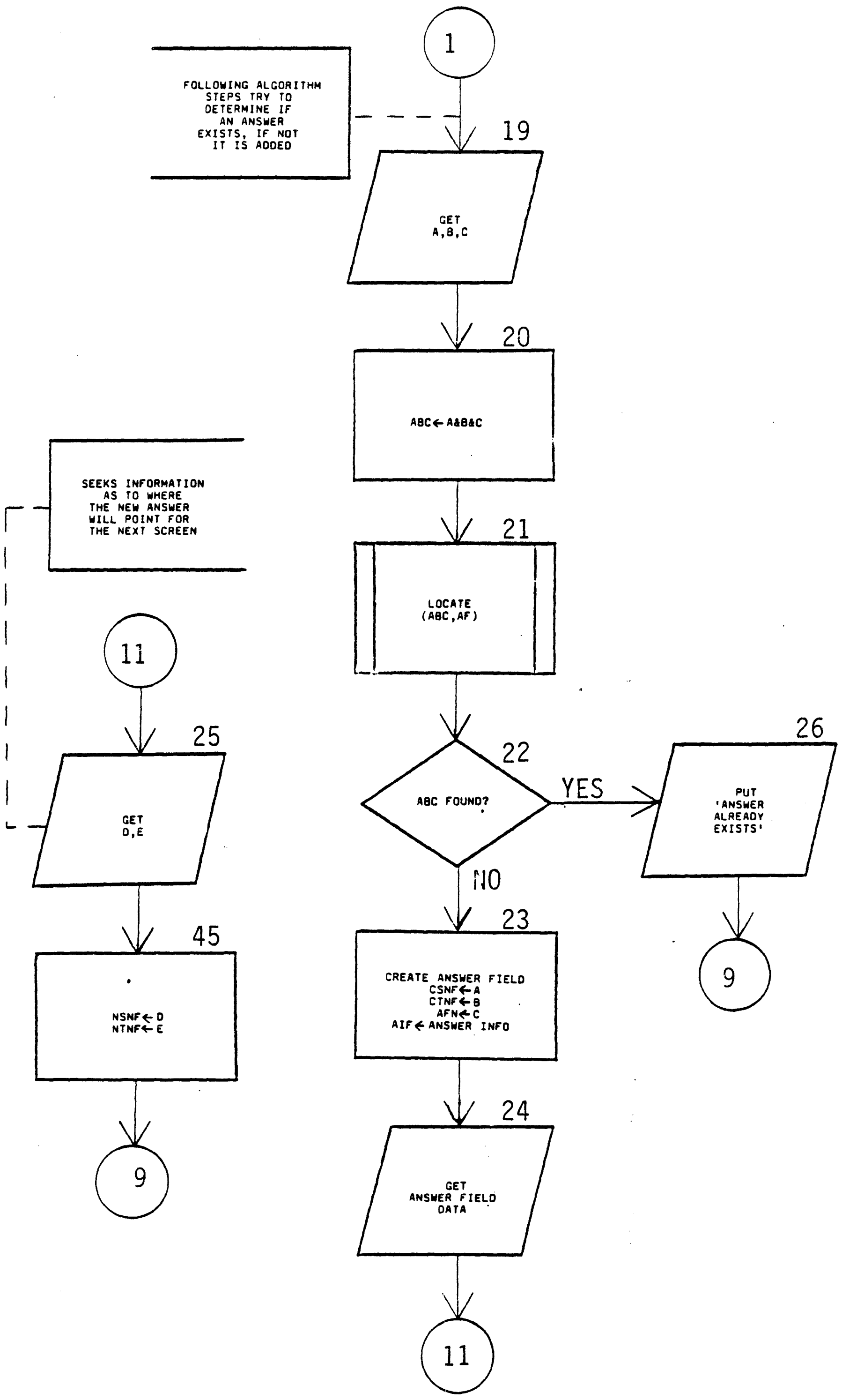

1
FOLLOWING ALGORITHM STEPS TRY TO DETERMINE IF AN ANSWER EXISTS, IF NOT IT IS ADDED
19
GET A,B,C
20
ABC← A&B&C
21
LOCATE (ABC,AF)
22
ABC FOUND?
YES
NO
26
PUT 'ANSWER ALREADY EXISTS'
9
23
CREATE ANSWER FIELD
CSNF← A
CTNF← B
AFN← C
AIF← ANSWER INFO
24
GET ANSWER FIELD DATA
11
SEEKS INFORMATION AS TO WHERE THE NEW ANSWER WILL POINT FOR THE NEXT SCREEN
11
25
GET D,E
45
NSNF← D
NTNF← E
9

Figure 29. (Continued)

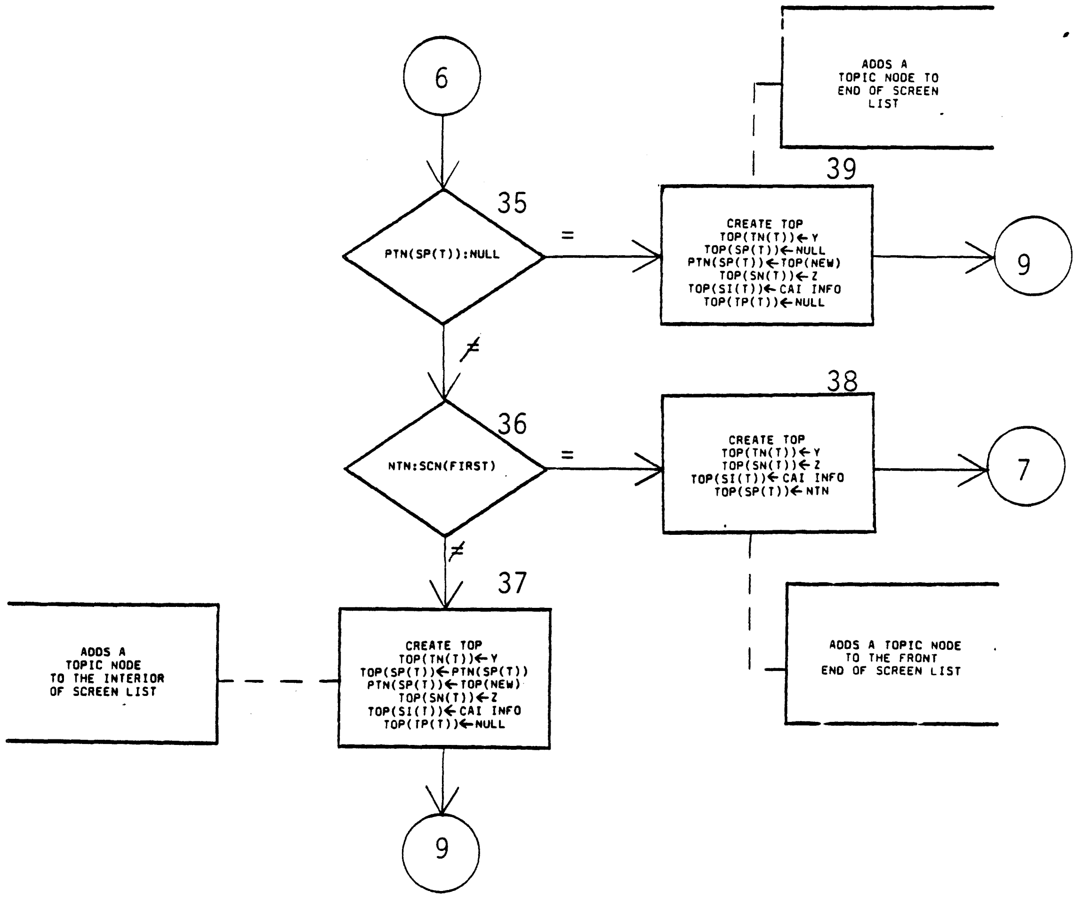


Figure 29. (Continued)

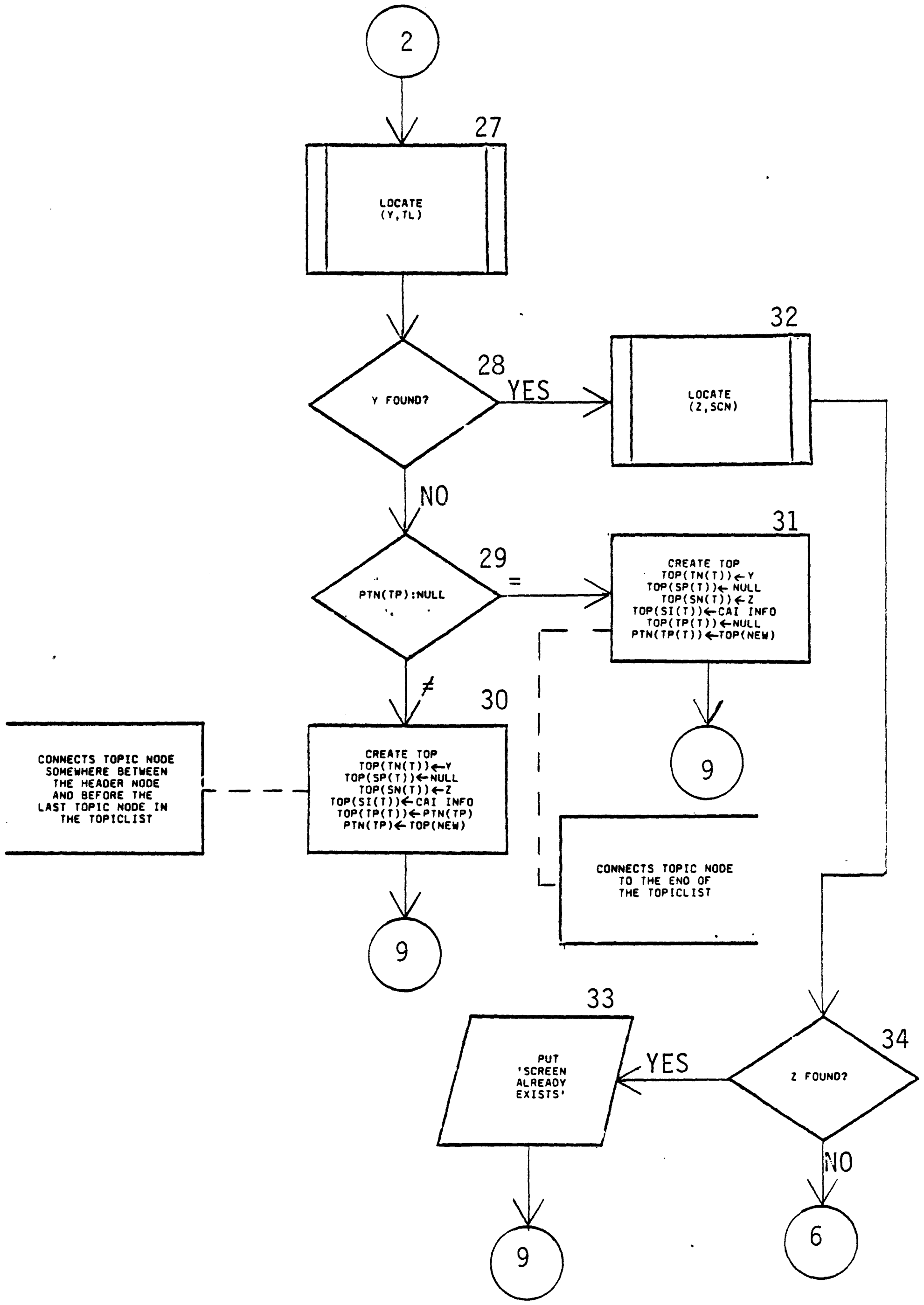


Figure 29. (Continued)

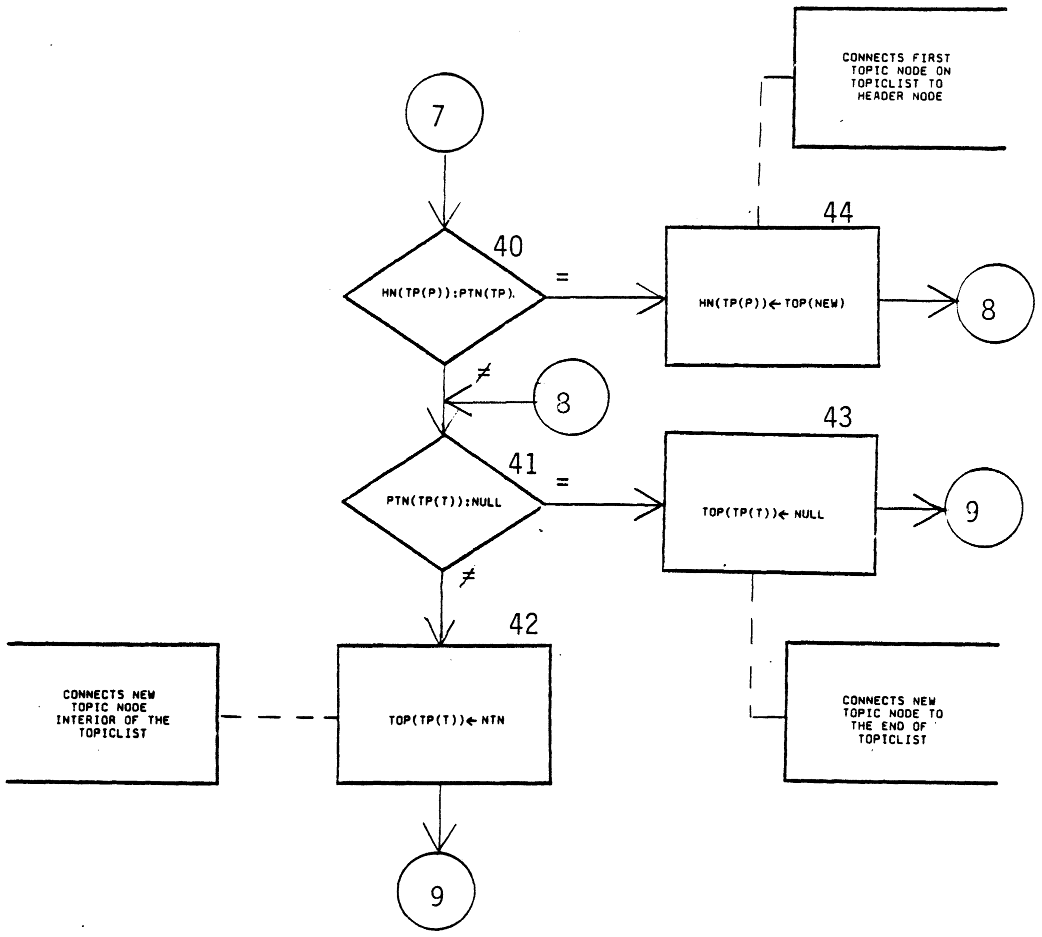

7
40
HN(TP(P)):PTN(TP).
=
≠
44
HN(TP(P))←TOP(NEW)
8
CONNECTS FIRST TOPIC NODE ON TOPICLIST TO HEADER NODE
8
41
PTN(TP(T)):NULL
=
≠
43
TOP(TP(T))← NULL
9
CONNECTS NEW TOPIC NODE TO THE END OF TOPICLIST
42
TOP(TP(T))← NTN
CONNECTS NEW TOPIC NODE INTERIOR OF THE TOPICLIST
9

Figure 29. (Continued)

Block 1: beginning of the Create/Add subprogram.

Block 2: an input/output block requesting information as to what area is to be manipulated, or do you want to quit.

Block 3: is a decision block to determine if the return mode was selected in order to terminate this function.

Block 4: wants to know if the answer file is what needs to be manipulated.

Block 5: input the subject, topic and screen identification numbers for the computer aided instruction display screen you want to create.

Block 6: call to subprogram routine described in Part 2 of this algorithm to see if X already exists on the subject list.

Block 7: was X found on the subject list?

Block 8: process node which creates a header node and sets the values and pointers to an initial position.

Block 9: decision block determines if this is the very first header node on the subject list.

Block 10: this decision block wants to isolate the new header node if it is to be added to the end of the subject list.

Block 11: analyzes if the new header node is to be added to the front of an existing subject list.

Block 12: process block which inserts the header node somewhere within the interior of the subject list.

Block 13: aligns the connectors to allow the header node to be inserted at the front of the subject list.

Block 14: moves the bottom-pointer to acknowledge the presence of another header node on the subject list and to align the downward pointer of the header node.

Block 15: this process block initializes the first node onto the subject list.

Block 16: terminator symbol signifying return from this function back to the main procedure.

Block 17: creates a topic node with the computer aided instruction screen information for any new header node just created on the subject list.

Block 18: realigns the connectors when adding a header node to the end of the subject list.

Block 19: after having selected the mode to create or add answers to the answer file, this input/output block requests information as to what subject, topic and answer identification numbers you desire to input a new piece of answer information.

Block 20: this process block concatenates the previous input identification numbers in a single number. This is done only to try and improve the search time required in the next block.

Block 21: call to a subprogram routine which determines if and where this identification number exists.

Blocks 22: was the identification number found in the answer file?

Block 23: says to create an answer structure in order to insert the reference field data into the answer node.

Block 24: requests the computer aided instruction answer be input so the answer information field can be completed.

Block 25: desires information from the instructor as to what display screen by subject and topic identification number follows this answer.

Block 26: display a message to user stating that answer already exists.

Block 27: call to an algorithm subprogram to determine if Y is on the topic list.

Block 28: question to determine if Y was found on topic list.

Block 29: decision block to determine the exact location of where a new topic node should be inserted, either in the interior or at the end of the topic list.

Block 30: process block which creates and connects topic node some where between the header node and before the last topic node on the topic list.

Block 31: creates a topic node and aligns the connectors for adding the topic node to the end of the topic list.

Block 32: call to a subprogram to determine if Z is on the screen list.

Block 33: produces a message to the user that the display screen already exists.

Block 34: was Z found on the screen list?

Block 35: wants to determine if the topic node to be added will go at the end of the screen list.

Block 36: analyzes the projected position of the new topic node to see if it should be the first topic node on the screen list.

Block 37: is the process which creates a new topic node and inserts it somewhere in the interior of the screen list.

Block 38: adds a topic node to the front of the screen list and realigns the connectors.

Block 39: creates a topic node and inserts it at the end of the screen list and carefully sets all connectors.

Block 40: asks the question if the new topic node is inserted between the header node and the current first topic node on the topic list.

Block 41: determines if the new topic node should be inserted at the end of the screen list or to the interior.

Block 42: connects the newly formed topic node somewhere to a location within the interior of the topic list.

Block 43: aligns the connectors of the previously formed topic node so it may reflect its insertion at the end of the topic list.

Block 44: establishs the connectors for a topic node inserted at the head of the topic list just after the header node.

Part 4 of the Assignment Presentation Algorithm

This part of the assignment presentation algorithm performs the function of deleting header nodes, topic nodes and answers, while at the same time realigning the flow of control connectors. It would seem a very trivial task to delete a node, and it is; however the resulting ramifications caused by the deletion are not so trivial of a situation. In this particular proposed algorithm, allowances must be made to determine the exact location of the topic node to be deleted. With its deletion, a determination must be made as to if only the screen list was affected or does it require a modification to the topic list also. This problem can percolate on up the subject list, and continue to the course origination point called the "head".

Carefully interpret the algorithm instructions within the following diagram. Several subtle linkages can be easily over looked.

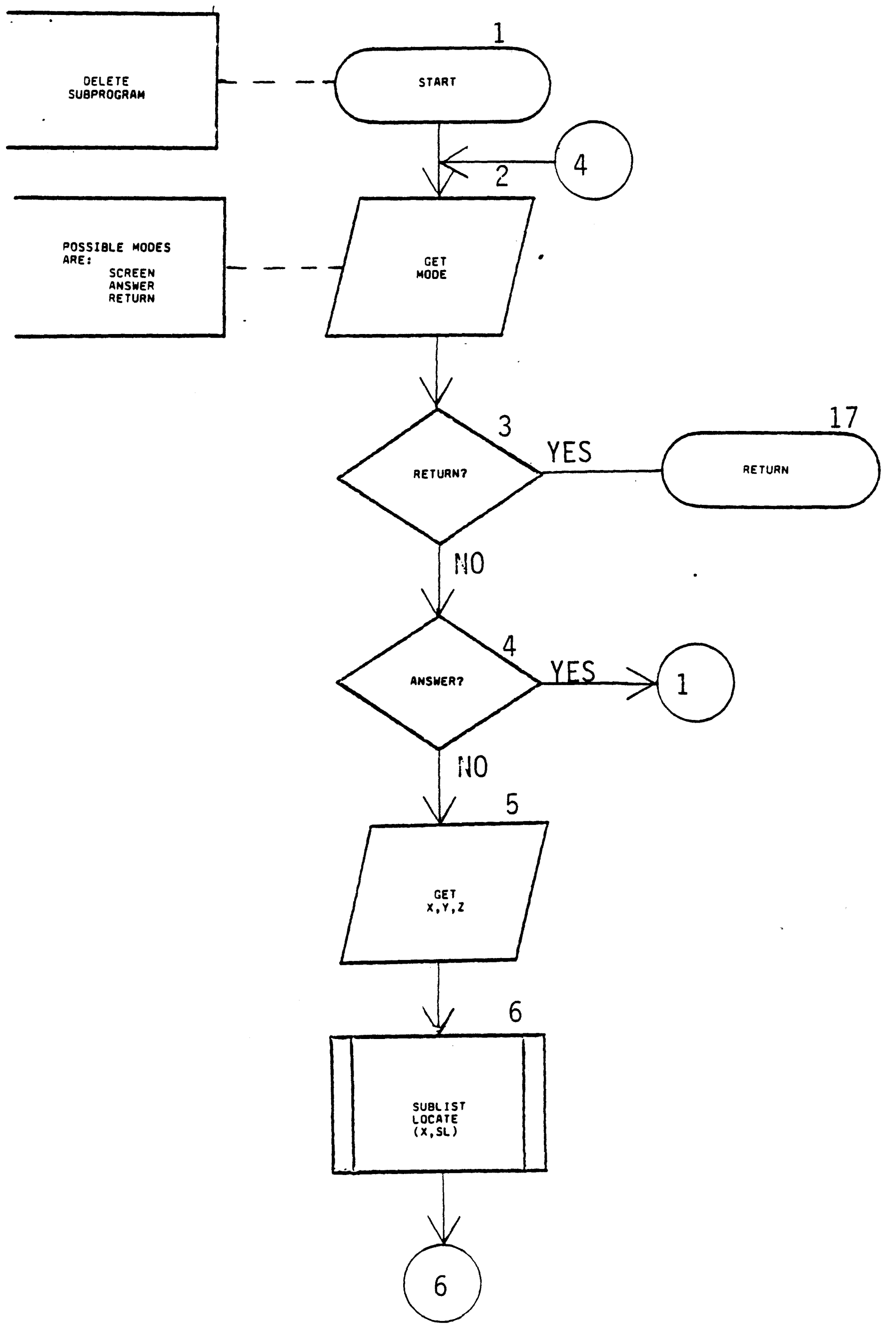


Figure 30. Delete Subprogram Algorithm

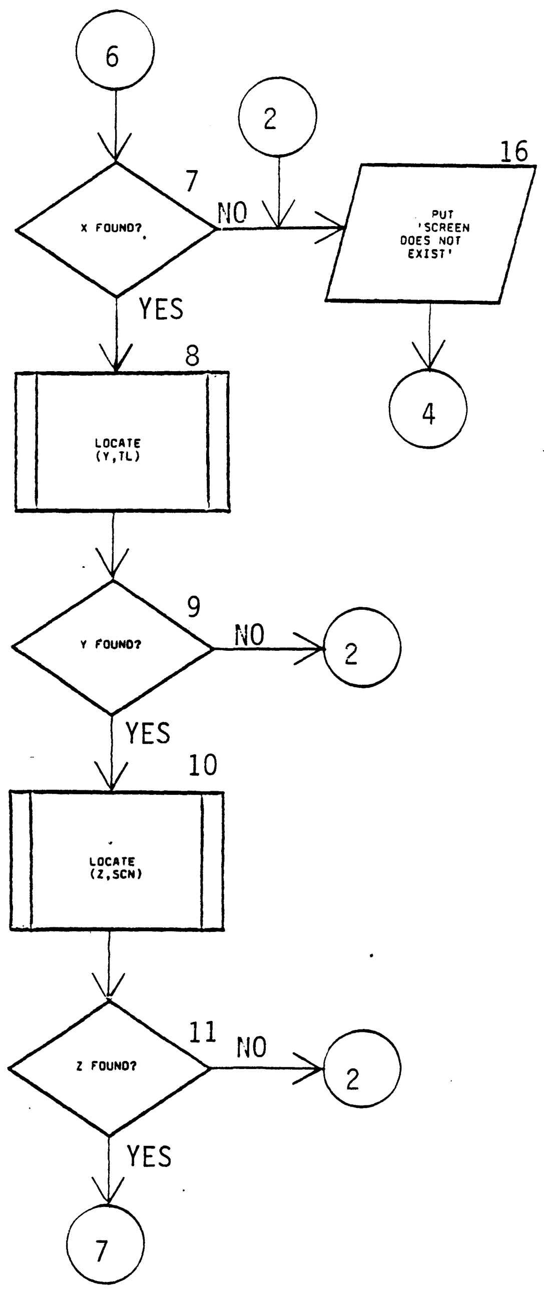


Figure 30. (Continued)

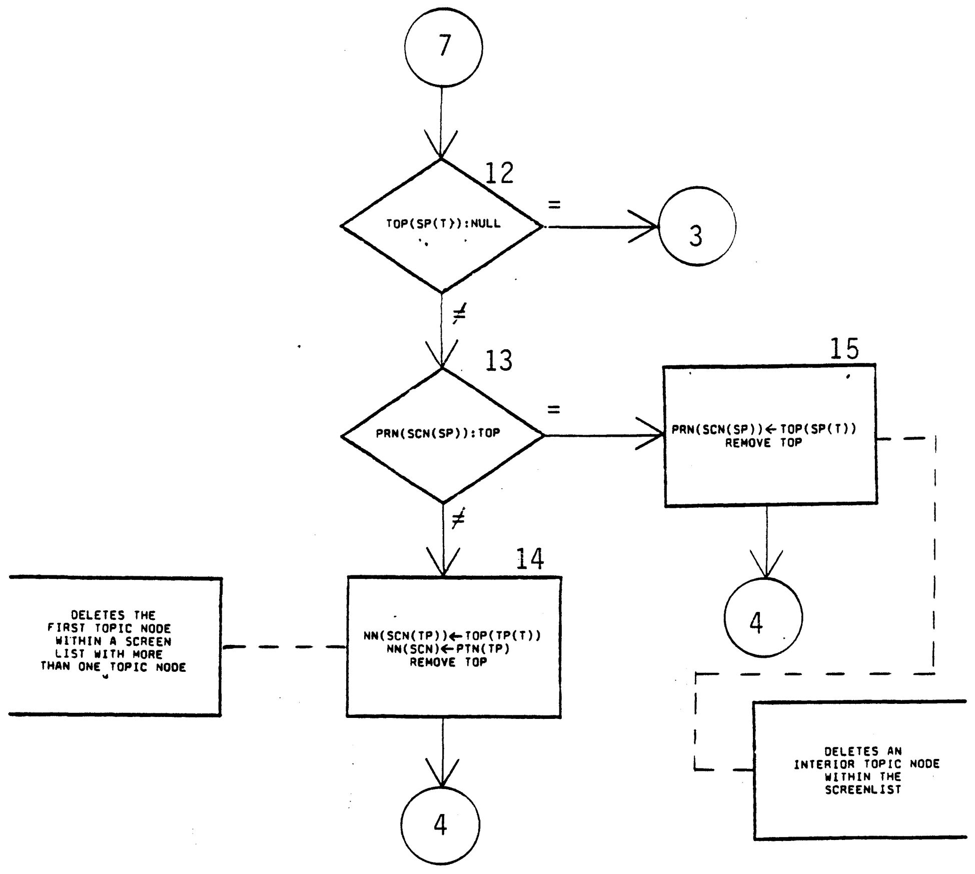

7
12
TOP(SP(T)):NULL
=
3
≠
13
PRN(SCN(SP)):TOP
=
15
PRN(SCN(SP))← TOP(SP(T))
REMOVE TOP
≠
14
NN(SCN(TP))←TOP(TP(T))
NN(SCN)←PTN(TP)
REMOVE TOP
DELETES THE FIRST TOPIC NODE WITHIN A SCREEN LIST WITH MORE THAN ONE TOPIC NODE
4
4
DELETES AN INTERIOR TOPIC NODE WITHIN THE SCREENLIST

Figure 30. (Continued)

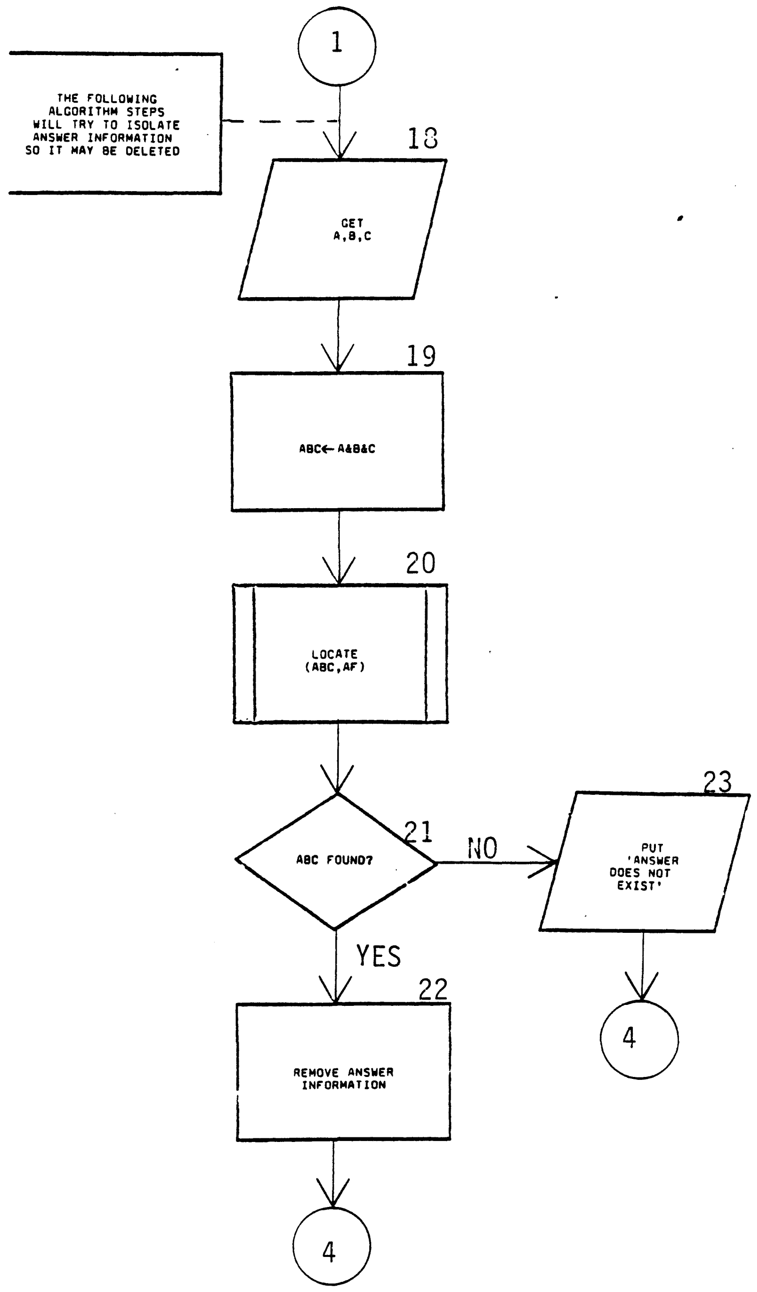
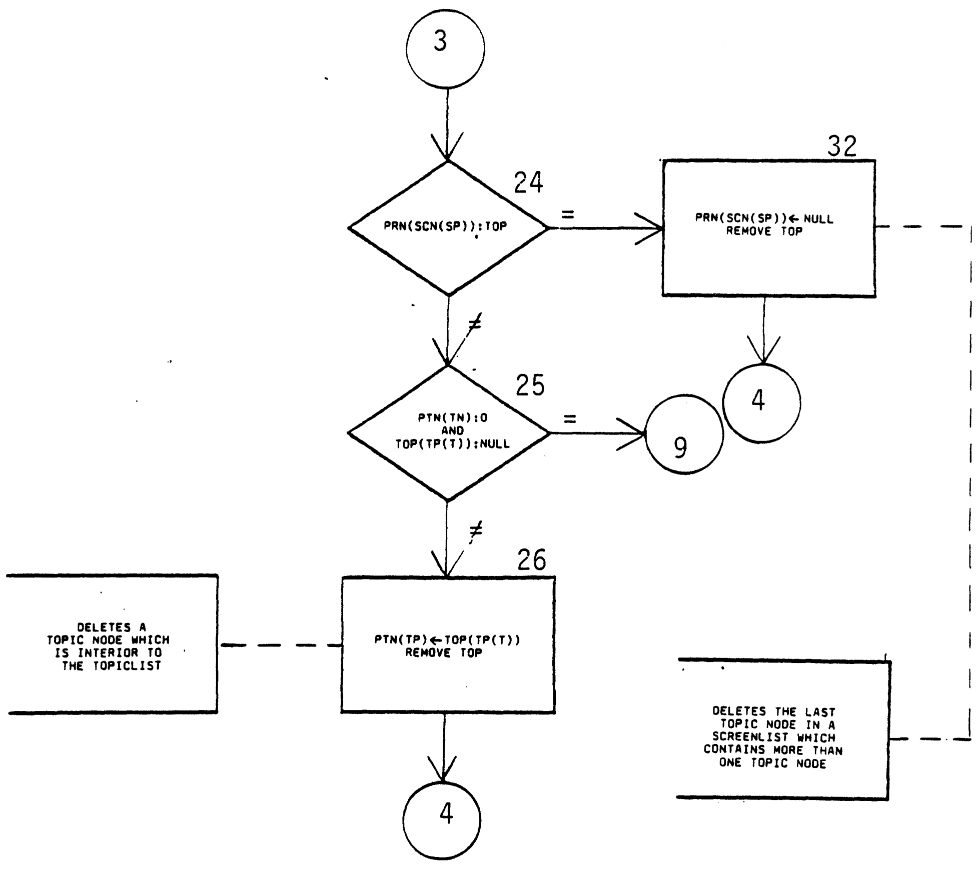

THE FOLLOWING ALGORITHM STEPS WILL TRY TO ISOLATE ANSWER INFORMATION SO IT MAY BE DELETED
1
18
GET A,B,C
19
ABC← A&B&C
20
LOCATE (ABC,AF)
21
ABC FOUND?
NO
23
PUT 'ANSWER DOES NOT EXIST'
4
YES
22
REMOVE ANSWER INFORMATION
4

Figure 30. (Continued)



3
24
PRN(SCN(SP)):TOP
=
32
PRN(SCN(SP))← NULL
REMOVE TOP
≠
4
25
PTN(TN):0
AND
TOP(TP(T)):NULL
=
9
≠
26
DELETES A TOPIC NODE WHICH IS INTERIOR TO THE TOPICLIST
PTN(TP)←TOP(TP(T))
REMOVE TOP
DELETES THE LAST TOPIC NODE IN A SCREENLIST WHICH CONTAINS MORE THAN ONE TOPIC NODE
4

Figure 30. (Continued)

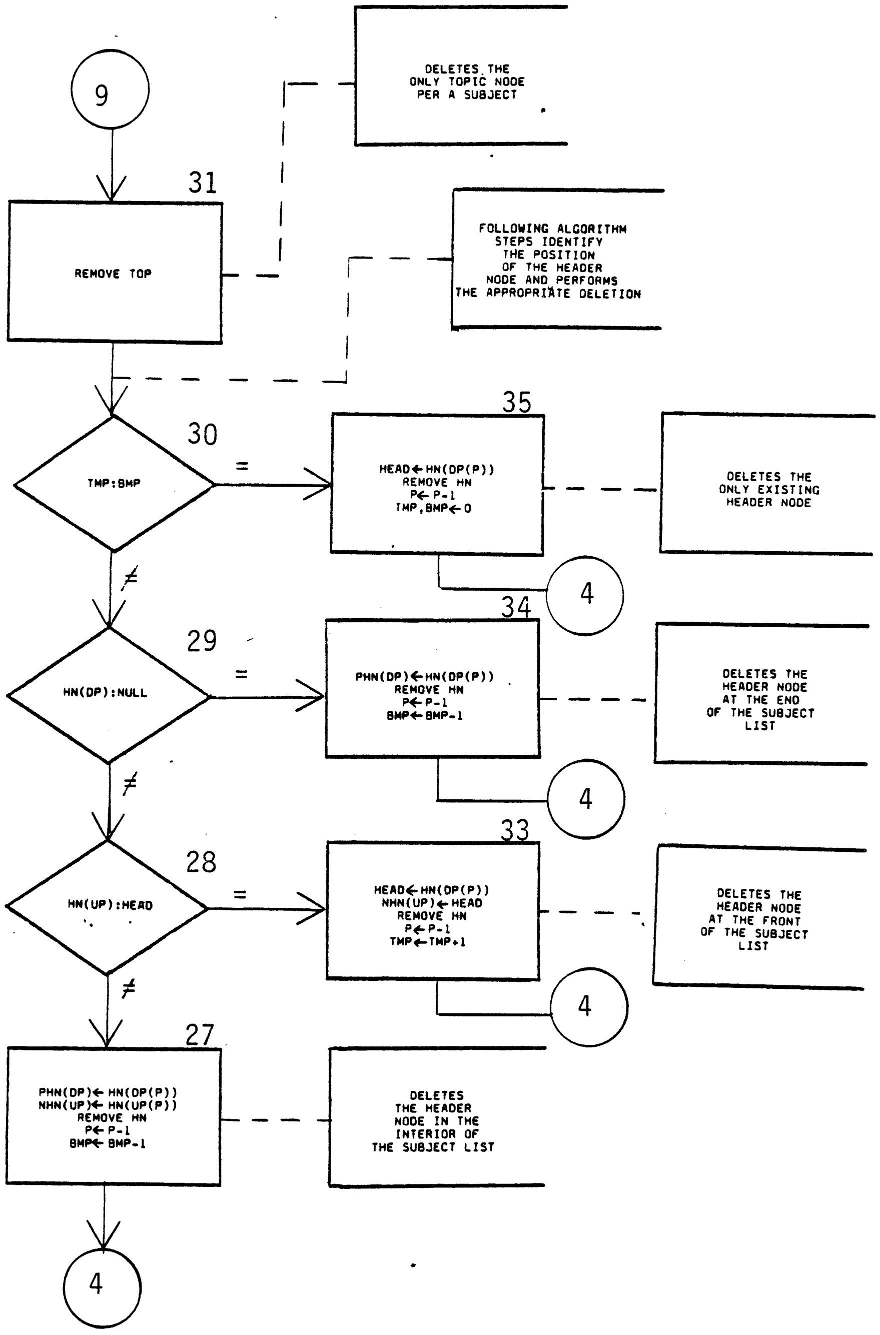

9
31
REMOVE TOP
DELETES THE ONLY TOPIC NODE PER A SUBJECT
FOLLOWING ALGORITHM STEPS IDENTIFY THE POSITION OF THE HEADER NODE AND PERFORMS THE APPROPRIATE DELETION
30
TMP:BMP
=
≠
35
HEAD← HN(DP(P)) REMOVE HN P← P-1 TMP,BMP← 0
DELETES THE ONLY EXISTING HEADER NODE
4
29
HN(DP):NULL
34
PHN(DP)← HN(DP(P)) REMOVE HN P← P-1 BMP← BMP-1
DELETES THE HEADER NODE AT THE END OF THE SUBJECT LIST
28
HN(UP):HEAD
33
HEAD← HN(DP(P)) NHN(UP)← HEAD REMOVE HN P← P-1 TMP← TMP+1
DELETES THE HEADER NODE AT THE FRONT OF THE SUBJECT LIST
27
PHN(DP)← HN(DP(P)) NHN(UP)← HN(UP(P)) REMOVE HN P← P-1 BMP← BMP-1
DELETES THE HEADER NODE IN THE INTERIOR OF THE SUBJECT LIST

Figure 30. (Continued)

Block 1: begin the deletion subprogram.

Block 2: acquire the appropriate mode for the particular action to be performed.

Block 3: was the "return" option selected?

Block 4: was the "answer option chosen in order to modify the answer file?

Block 5: input the subject , topic and screen number of the computer aided instruction display screen to be eliminated.

Block 6: call subprogram to determine if the subject is on the subject list.

Block 7: was the subject found on the list?

Block 8: call subprogram to determine if the topic desired exists.

Block 9: was the desired topic found?

Block 10: call the subprogram to find out if the screen desired exists.

Block 11: was the screen found on the screen list?

Block 12: decide if the screen desired is in the last topic node of the screen list.

Block 13: determine if the location of the topic node is in the interior or at the first of the screen list.

Block 14: process which deletes the first topic node within the screen list if it contains more than one topic node.

Block 15: the actions performed to delete a topic node located in the interior of the screen list.

Block 16: output a message that it is impossible to delete a screen that does not exist.

Block 17: causes the exit from this subprogram back to the calling algorithm.

Block 18: needs input concerning the subject, topic and answer numbers of the desired answer field to be deleted.

Block 19: concatenates the three previous input values.

Block 20: calls a subprogram to perform a search to determine if this particular answer exists.

Block 21: was the answer found to exist?

Block 22: delete answer information.

Block 23: answer does not exist, therefore it can not be deleted.

Block 24: determines if the topic node desired is the last node in the screen list.

Block 25: determines if the sought after topic node is the only topic node in the topic list.

Block 26: this process block deletes a topic node which is located in the interior of the topic list.

Block 27: this process deletes the header node in the interior of the subject list which no longer has any topic nodes associated with it.

Block 28: decision block determining if the header node to be deleted is the first on the subject list.

Block 29: determines if the header node to be deleted is in the last position of the subject list.

Block 30: determines if the header node about to be eliminated is the only header node on the subject list.

Block 31: delete the topic node.

Block 32: deletes the last topic node in a screen list which contains more than one topic node.

Block 33: deletes the header node at the front of the subject list.

Block 34: Deletes the header node at the end of the subject list.

Block 35: process to delete the only existing header node on the subject list.

Summary

This chapter has presented the assignment presentation algorithm module which is the first part of the computer aided instruction package being formulated. The overall design concept has been illustrated and the rational of what each block of the algorithm is performing has been given.

Appendix A at the end of this study provides several examples of screen implementation situations which exercises the aforementioned algorithm. Each exercised example represents critical points within this algorithm of the nature which could produce catastrophic affects if not properly performed.

CHAPTER V

RESPONSE EVALUATION ALGORITHM

Introduction

This chapter attempts to describe the algorithm design of a computer aided instruction module which will serve as a template for student response evaluation while interactively participating in an on-line course of instruction. Discussion will be focused on the design concept and how the algorithm functions.

General Design Concept

The general design concept of the Response Evaluation Algorithm can be visualized as the procedure which executes the computer aided instruction display screens and evaluates a student's responses in relation to requests put forward by the block of instruction. After evaluating a user's response, the automated course displays an appropriate comment or answer, before proceeding to the next display screen of information based off the user's previous response. Figure 31 illustrates the general conceptual flow of the response evaluation algorithm.

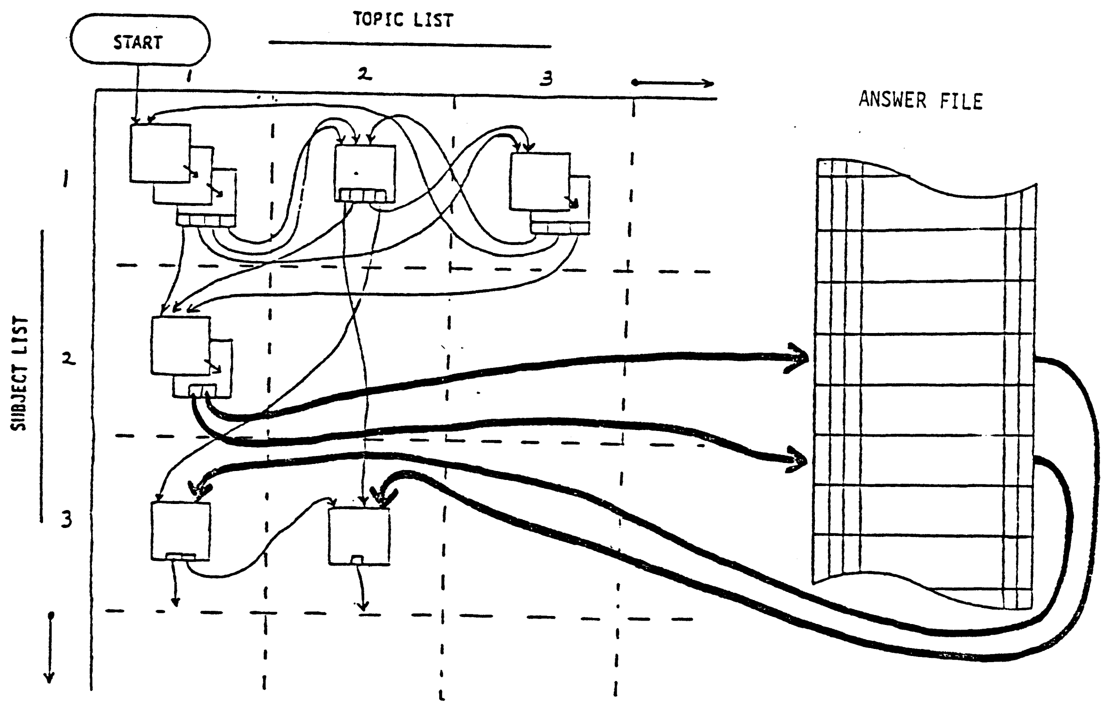


Figure 31. Conceptual View of Response Evaluation Algorithm

Now suppose a subject topic presentation consists of more than one screen's worth of information which must be shown to the student. In this case, it is assumed the student will be informed to press the "enter" key or some other defined operator in order to advance to the next display screen of information. Refer to the area encapsulated by the dotted box in Figure 31 which portrays this flow sequence.

Algorithm Development

In developing the response evaluation algorithm to show the sequenced flow of control during execution of the course of instruction, an initial point of debarkation must be established. For this algorithm the initial display screen of computer aided instruction material begins at subject and topic grid positions one. In reality this initial point could be a menu screen of some type telling a user what to do, or where to begin. But for this study this trivial point is bypassed and left to the discretion of the algorithm's implementor.

This algorithm uses the same terminology as the previously mentioned assignment presentation module. It uses the assignment presentation package as a building block for the initialization required to execute this module. The control flow of this proposed design should prove quite flexible to a wide variant of implementations. Figure 32 reveils the response evaluation algorithm.

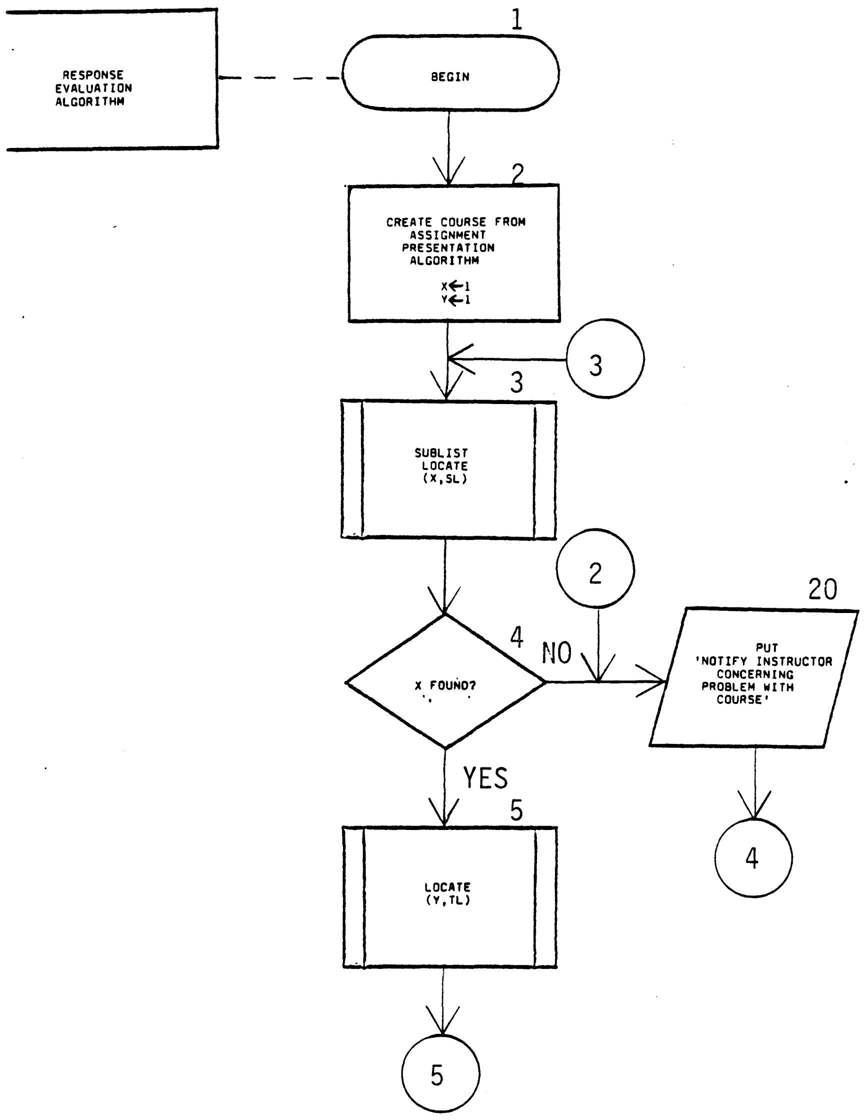

RESPONSE EVALUATION ALGORITHM
1
BEGIN
2
CREATE COURSE FROM ASSIGNMENT PRESENTATION ALGORITHM
X←1
Y←1
3
3
SUBLIST LOCATE (X,SL)
2
20
4
NO
X FOUND?
PUT 'NOTIFY INSTRUCTOR CONCERNING PROBLEM WITH COURSE'
YES
5
LOCATE (Y,TL)
4
5

Figure 32. Response Evaluation Algorithm

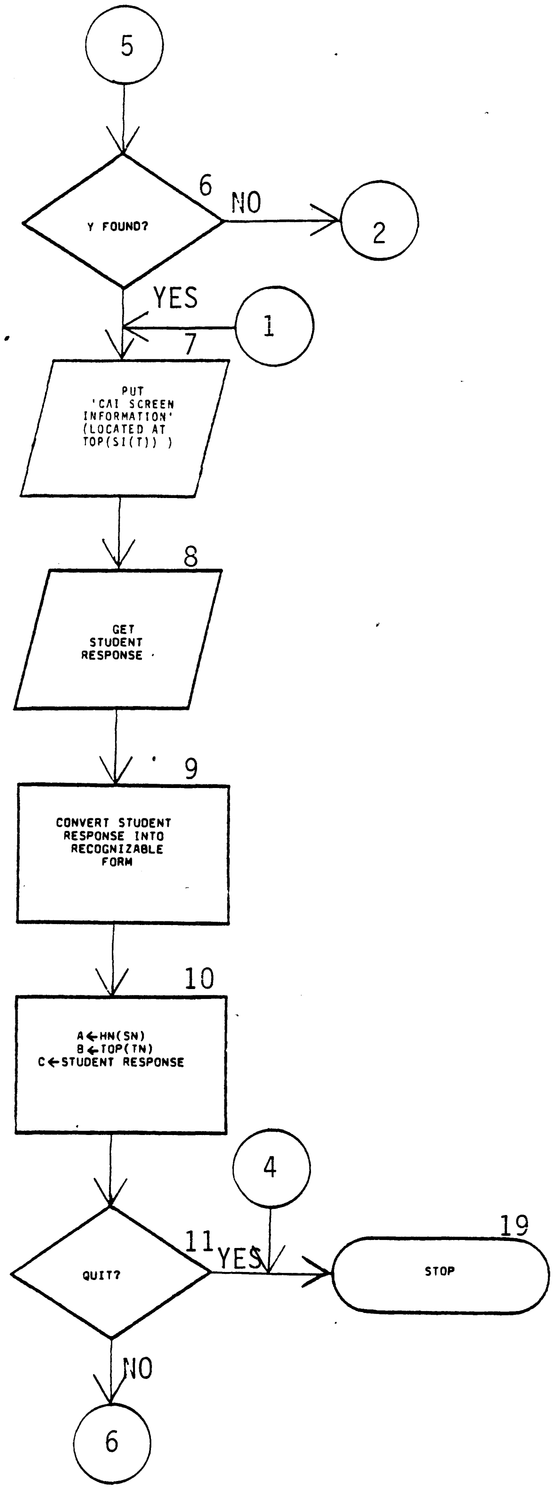

5
6
NO
Y FOUND?
2
YES
1
7
PUT 'CAI SCREEN INFORMATION' (LOCATED AT TOP(SI(T)))
8
GET STUDENT RESPONSE
9
CONVERT STUDENT RESPONSE INTO RECOGNIZABLE FORM
10
A←HN(SN)
B←TOP(TN)
C←STUDENT RESPONSE
4
11
YES
19
QUIT?
STOP
NO
6

Figure 32. (Continued)

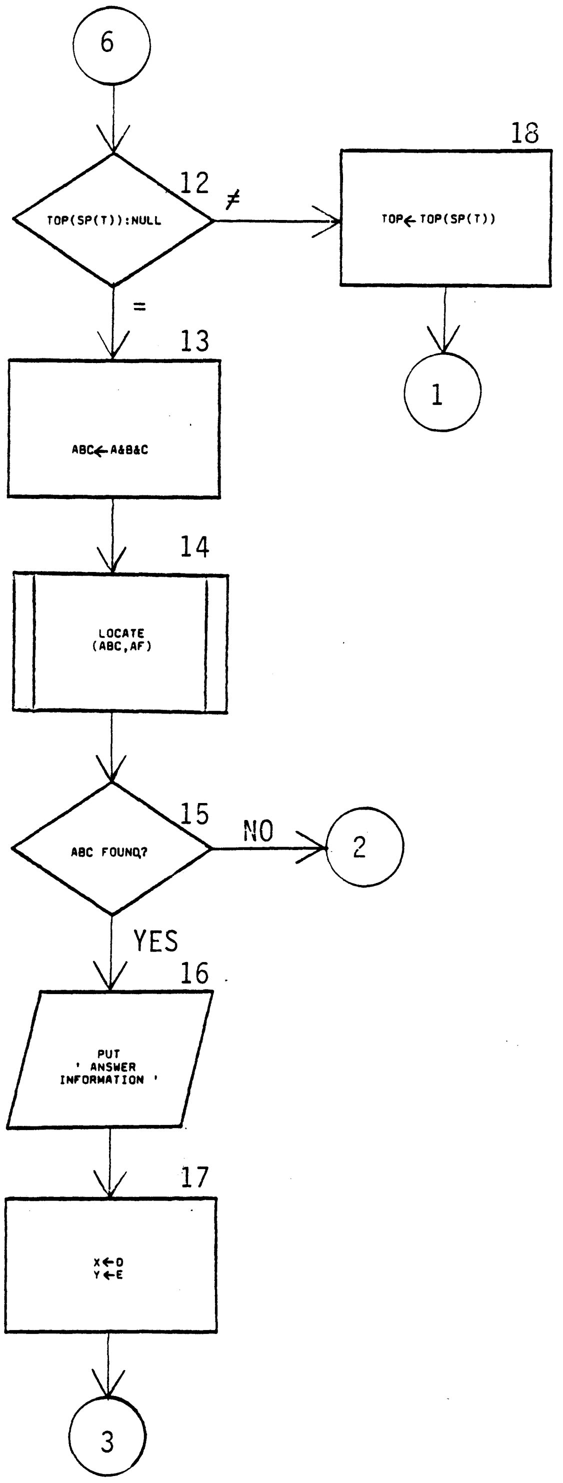


Figure 32. (Continued)

Conceptually, this algorithm displays a screen of information, acquires a response from the user, extrapolates and interprets the meaning of the user's response, performs an action or displays an answer, and then repeats this process until discontinued.

As with the previous assignment presentation module, lets walk through this algorithm by the numbers in order to obtain an understanding of what is taking place.

Block 1: begins the response evaluation algorithm.

Block 2: signifies a process which initializes or creates the course from the various input files or data sets into a structure which can be visualized like Figure 3. For instructional purposes only this algorithm course begins with the first display screen located at subject 1 and topic 1 (matrix position 1, 1). X & Y are variables which receive the values accordingly.

Block 3: is a subprogram call to a routine to determine the location of X on the subject list.

Block 4: was X found on the subject list?

Block 5: subprogram call to determine the position of Y on the topic list. The identification of this position on the topic list is the location of the first display screen for this item of information.

Block 6: was Y found on the topic list?

Block 7: display the computer aided instruction text screen located at this position.

Block 8: acquire user's response.* ("*" means this subject will be discussed more in depth at the end of this algorithm's walk through)

Block 9: this process block signifies the user's response will be interpreted from whatever form it maybe input, into a recognizable form which can specify a unique answer in the answer file based off the subject and topic identifications.*

Block 10: assigns the subject number, topic number and determined student response to variables for later manipulation.*

Block 11: was the understood response to discontinue execution of the course.

Block 12: determines if the display screen previously shown the user was in the last topic node of the screen list.

Block 13: concatenates the subject, topic and answer values preparing for an up coming search for the value in the answer list.

Block 14: subprogram call to determine location of answer sought as a response to the user's previous input.

Block 15: was this position found on the answer list.

Block 16: display the answer to the user.

Block 17: acquire from the answer structure the location of the next computer aided instruction display screen in which to show and assign those values to the X and Y variables.

Block 18: is the process which increments to the next display screen on the screen list.

Block 19: terminates the response evaluation algorithm.

Block 20: print a message to the user the response evaluation module can not find the next display screen of information, and that the program contains a flaw. This condition should not occur, but if it does it must be brought to the attention of the instructor immediately.

Blocks 8, 9 and 10 of this module were annotated with a star (*) to signify the possible expansion of these blocks to incorporate additional information when this algorithm is implemented. Block 8 of this module specifies an input will be received from the student user. The context of this input could be in the form of a standard response to finite number of possible answers (i.e., a question has four possible answers: A, B, C, or D), or could be a response expressed by a word, phrase, sentence, or a picture. It is not to difficult to see, that if the answer could take on a variable posture of not being a specific answer that the input could become quite entailed. This brings about the importance of Block 9 which is a process which puts the response in a recognizable form. If the instruction system posed a question with four possible answers (A, B, C, D), then the algorithm implementor may want to create a symbol table in order the response evaluation system could understand that a input of the character "1" would mean the same as the required answer of "A", and so forth. If however, the answer required was more of an obscure nature, then the implementor may want to incorporate a natural language processor module in order to determine the intent of the user's input. An example of such a process could be a question which asks for a phrase to be written in another language, like German. This phrase would have to undergo close scrutiny in order to determine if the question was satisfactorily answered since there may be several correct answers. it is at this point that Block 10 becomes important. Block 10 takes evaluated user responses from the previous block and establishes the location of the specified answer for the previous question. Block 10 also is where a historical student response list would be updated in order to provide statistical information as to a students progress and /or areas needing additional emphasis because of poor performance.

Summary

The algorithm described within this chapter portrays a template for creating an interactive student response evaluation system. It illustrated the overall design concept and reveiled areas of possible expansion based upon the needs and expertise of the implementor.

CHAPTER VI

SUMMARY AND FUTURE WORK

Summary

Purpose of this study was to create a algorithm package consisting of a computer aided instruction assignment presentation and response evaluation system. The design serves as a template for experienced programmers developing their own computer aided instruction presentation systems.

Descriptions of this algorithm package are illustrated using the standard flowcharting symbols accompanied by pseudo-English statements representing specific actions taking place in a particular order. Implementations of this algorithm design will produce a computer aided instruction course capable of further expansion depending upon the programming experience of the implementor.

The assignment presentation system algorithm portrays the linkage required for an instructor to be able to create/add, edit/locate, and delete course presentation screens and answers. It handles the special conditions of manipulating screens at the beginning, end, and interior of the presentation text. The physical arrangement of answers on the answer list and the mapping mechanism for accessing them allows for an infinite (in theory) number of response possibilities.

The response evaluation module is a very flexible algorithm which allows many variations, from a very controlled student response to a very abstract response scheme utilizing state of the art natural language processing.

Future Work

Availability of future applications surrounding the area of computer aided instruction is only restricted by one's imagination. Technology is changing so fast that ideas are barely keeping ahead of physical applications. Proposed areas of further research associated with the software aspects of computer aided instruction are:

Creation of more versatile software capable of analyzing and coordinating actions facilitating computer aided instruction with video disks, lasers, satellites, remote sensing and oral communication devices.

Research into the development of better fault tolerant modes of instructional presentation systems.

Development of advanced authoring systems which utilize artificial intelligence.

Exploratory research into the capability of allowing students to ask questions of the computer while actively engaged in some form of computer aided instruction.

Algorithm research into the creation of new storage compaction techniques specifically oriented toward computer aided instruction presentations.

BIBLIOGRAPHY

Aho, Alfred V., Hopcroft, John E., and Ullman, Jeffery D. Data Structures And Algorithms. Reading, MA: Addison-Wesley Publishing Company, 1983.

Aiken, Robert, and Braun, Ludwig. "Into The 80's With Computer-Based Learning." COMPUTER 13, 7 (1980).

Alessi, Stephen M., and Trollip, Stanley R. Computer-Based Instruction. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1985.

Anderson, Ronald, and Klassen, Daniel. "A Conceptional Framework For Developing Computer Literacy Instruction." AEDS Journal 14, 3 (1981), 128.

Atkinson, R. C., and Wilson, H. A. "Computer Assisted Instruction." Science. 162, 1968, 73-77.

Baase, Sara. Computer Algorithms. Reading, MA: Addison-Wesley Publishing Company, 1978.

Ballaben, G. and Ercoli, P. "Computer-Aided Teaching of Assembler Programming." In O. Lecarme and R. Lewis (Eds.), Computers in Education , IFIP (Part 1). Amsterdam: North-Holland, 1975, 217-227.

Barr, A., Beard, M., and Atkinson, R. C. "A Rationale and Description of a CAI Program to Teach the BASIC Programming Language." Instructional Science 4 (1975), 1-31.

Barr, A., Beard, M., and Atkinson, R. C. "Information Networks for CAI Curriculums." In O. Lecarmi and R. Lewis (eds.), Computers in Education , IFIP (Part 1). Amsterdam: North-Holland, 1975, 477-582.

Barr, A., Beard, M., and Atkinson, R. C. "The Computer as a Tutorial Laboratory: The Stanford BIP Project." International Journal of Man-Machine Studies 8 (1976), 567-596.

Bayman, P. and Mayer, R. E. "A Diagnosis of Beginning Programmers' Misconceptions of BASIC Programming Statements." Communications of the ACM 26, 9 (1983), 677-679.

Bitter, Gary G., and Camuse, Ruth A. Using A Microcomputer In The Classroom. Reston, VA: Reston Publishing Company, Inc., 1984.

Blaschke, C. L. "Microcomputer Software Development for Schools: What, Who, How?" Educational Technology 19, 10 (1979), 26-28.

Bork, A. Learning With Computers. Bedford, MA: Digital Press, 1981.

Bork, A. and Franklin, S. "Personal Computers in Learning." Educational Technology 19, 10 (1979), 7-12.

Brown, J. S., Burton, R. R., and Bell, A. G. "SOPHIE: A Step Toward Creating a Reactive Learning Environment." International Journal of Man-Machine Studies 7 (1975), 675-696.

Brown, P. J., Griffiths, M., Griswold, R. E., Lawson, H. W., Niblett, B., Richards, M., Spratt, E. B., Waite, W. M., and Wichmann, B. A. Software Portability. London, England: Cambridge University Press, 1977.

Burke, Robert L. CAI Sourcebook. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1982.

Calvin, Allen D. Programmed Instruction. Bloomington, IN: Indiana University Press, 1969.

Carbonell, J. "AI in CAI: An Artificial-Intelligence Approach to Computer-Assisted Instruction." IEEE Transactions on Man-Machine Systems MMS-11, 4 (1970), 190-202.

Chambers, J. A. and Sprecher, J. W. "Computer Assisted Instruction: Current Trends and Critical Issues." Communications of the ACM 23, 6 (1980), 332-342.

Chan, Julie M. T., and Korostoff, Marilyn. Teachers' Guide To Designing Classroom Software. Beverly Hills, CA: SAGE Publications, Inc., 1984.

Clancey, William J. "Tutoring Rules For Guiding A Case Method Dialogue." Intelligent Tutoring Systems. New York, NY: Academic Press, Inc., 1982, 201-226.

Coburn, Peter, Kelman, Peter, Roberts, Nancy, Snyder, Thomas F. F., Watt, Daniel H., and Weiner, Cheryl. Practical Guide To Computers In Education. Reading, MA: Addison-Wesley Publishing Company, 1982.

Collins, William J. "A Data-Oriented Introduction To Program Design." SIGCSE BULLETIN 11, 4, Dec (1979), 49-55.

Computer-Assisted Instruction Guide. 2 Ed. Newbury, MA: ENTELEK, Inc., 1968.

Cox, Margaret J. "CAL In A Changing Curriculum." Computer Education Jun (1982), 24-25.

Critchfield, M. "Beyond CAI: Computers as Personal Intellectual Tools." Educational Technology 19, 10 (1979), 18-25.

Curtin, Constance. "Publication Practices For Microcomputer Programs." Journal Of Computer-Based Instruction 7, 4, 123.

Davies, Colin S. "Peer Tutors: Their Utility And Training In The Personalized System Of Instruction." Educational Technology 11 (1978), 23-25.

Dean, C. and Whitlock, Q. A Handbook of Computer Based Training. New York: Nichols Publishing Co., 1983.

Dijkstra, Edsger W. A Discipline of Programming. Englewood Cliffs, NJ: Prentice Hall, Inc., 1976.

Dimas, C. "A Strategy for Developing CAI." Educational Technology 18, 4 (1978), 26-29.

Dromey, R. G. How to Solve It by Computer. London: Prentice/Hall International, Inc., 1982.

Ellis, Allen B. The Use & Misuse Of Computers In Education. Newton, MA: McGraw-Hill, Inc., 1974.

Endsley, William R. PEER Tutorial Instruction. Englewood Cliffs, NJ: Educational Technology Publications, Inc., 1980.

Fine, Benjamin. Teaching Machines. New York, NY: Sterling Publishing Company, Inc., 1962.

Foltz, Charles I. The World of Teaching Machines. Washington, DC: Electronic Teaching Laboratories, 1961.

Gagne, Robert M., Wager, Walter, and Rojas, Alicia. "Planning And Authoring Computer-Assisted Instruction Lessons." Educational Technology 9 (1981), 17-21.

Garson, James W. "Getting Problem-Solving Advice From A Computer." BYTE 6, 5, May (1981), 186.

Gauthier, Richard, and Ponto, Stephen. Designing Systems Programs. Englewood Cliffs, NJ: Prentice Hall, Inc., 1970.

Goldberg, A. and Suppes, P. "A Computer-Assisted Instruction Program for Exercises on Finding Axioms." Educational Studies in Mathematics 4 (1972), 429-549.

Goldstein, I. I. Training: Program Development and Evaluation. Monterey, CA: Brooks/Cole, 1974.

Goodman, S. E., and Hedetniemi, S. T. Introduction To The Design And Analysis of Algorithms. New York, NY: McGraw-Hill, Inc., 1977.

Gray, D. C., Hulskamp, J. P., Kumm, J. H., Lichtenstein, S., and Nimmervoll, N. E. "COAL-A - A Minicomputer CAI System." IEEE Transactions on Education E-20, 1 (1977), 73-77.

Harper, Dennis O., and Stewart, James H. RUN: Computer Education. Belmont, CA: Wadsworth, Inc., 1983.

Hazen, Margret. "Computer-Assisted Instruction With PILOT On The Apple Computer." Educational Technology 11 (1982), 20-22.

Heines, Jesse M. "Courseware Development And The NSF." COMPUTER 13, 7 (1980), .

Heines, J. M. Screen Design Strategies for Computer-Assisted Instruction. Bedford, MA: Digital Press, 1984.

Hickey, Albert E. Computer-Assisted Instruction: A Survey Of The Literature. Newburyport, MA: ENTELEK, Inc., 1968.

Hoare, C. A. R. "An Axiomatic Basic For Computer Programming." Communications of the ACM. 12, 10 (1969), 576-583.

Hofmeister, Alan. Microcomputer Applications In The Classroom. New York, NY: CBS College Publishing, 1984.

Holmes, Glyn. "Computer-Assisted Instruction: A Discussion Of Some Of The Issues For Would-Be Implementors." Educational Technology 9 (1982), 7-13.

Jones, Bush. "Teaching Algorithm Design." SIGCSE BULLETIN 11, 4, Dec (1979), 27-30.

Jurgemeyer, Fred H. "Programmed Instruction: Lesson It Can Teach Us." Educational Technology 5 (1982), 20-21.

Katz, M. R. and Chapman, W. "SIGI: An Example of Computer-Assisted Guidance." Educational Technology

Keeton, Roy "Teaching Data Processing By Case Study." Computer Education 11 (1981), 2-7.

Kehler, T. P. and Barnes, M. "Design for an On-Line Consultation System." AEDS Journal 14, 3 (1981), 113-127.

Knuth, Donald E. The Art Of Computer Programming. 1, Reading, MA: Addison-Wesley Publishing Company, 1968.

Koffman, E. B. and Blount, S. E. "Artificial Intelligence and Automatic Programming in CAI." Artificial Intelligence 6 (1975), 215-234.

Lagowski, J. J. "Computer-Assisted Instruction in Chemistry." In W. H. Holtzman (ed.), Computer-Assisted Instruction, Testing, and Guidance. New York: Harper & Row, 1970, 283-298.

Lahey, George. "The Effect Of Instructional Sequence On Performance In Computer-Based Instruction." Journal Of Computer-Based Instruction 7, 4, 111.

Lantz, B. S., Bregar, W. S., and Farley, A. M. "An Intelligent CAI System for Teaching Equation Solving." Journal of Computer-Based Instruction 10, 1 & 1 (1983), 35-52.

Lathrop, Ann, and Goodson, Bobby. Courseware In The Classroom. Reading, MA: Addison-Wesley Publishing Company, Inc., 1983.

Levien, R. E., and Mosmann, C. "Instructional Uses of Computers." The Emerging Technology. New York, NY: The Rand Corporation, 51-82.

Lorton, P. Jr. and Cole, P. "Computer-Assisted Instruction in Computer Programming: SIMPLER, LOGO, and BASIC, 1968-1970." In P. Suppes (ed.), University-Level Computer-Assisted Instruction at Stanford: 1968-1980. Stanford, CA: Stanford University, Institute for Mathematical Studies in the Social Sciences, 1981, 841-876.

McGowan, Clement L., and Kelly, John R. Top-Down Structured Programming Techniques. London, England: Mason/Charter Publishers, Inc., 1975.

McIsaac, Donald N., and Baker, Frank B. "Computer-Managed Instruction System Implementation On A Microcomputer." Educational Technology. 10 (1981), 40-46.

Mc Vay, P. O. "Tapping The Appeal Of Games." PIPELINE Spring (1981),8.

Magidson, E. M. "Issue Overview: Trends in Computer-Assisted Instruction." Educational Technology 18, 4 (1978), 5-8.

Matthews, J. I. "Microcomputer vs. Minicomputer for Educational Computing." Educational Technology 18, 11 (1978), 19-22.

Maynard, J. Modular Programming. Princeton: Auerbach Publishers, 1972.

Meredith, J. C. The CAI Author/Instructor. Englewood Cliffs, NJ: Educational Publications, Inc, 1971.

Merrill, M. D., Schneider, Edward W., and Fletcher, Kathie A. TICCIT. Englewood Cliffs, NJ: Educational Technology Publications, Inc., 1980.

Miller, Lance A., and Thomas, John C. "Behavioral Issues In the Use of Interactive Systems." International Journal of Man-Machine Studies. 9, 1977, 509-536.

Mitzel, Harold E. "On The Importance Of Theory In Applying Technology To Education." Journal Of Computer-Based Instruction 7, 4, 93.

Morris, John M. "The Case For CAI." SIGCUE Bullentin. Winter (1984), 11-14.

Morris, Judith. "Questions About Computer Based Learning." Computer Education Nov (1981), 26-28.

PIPELINE. "Are We Ready For Computer-Assisted Instruction?" PIPELINE Spring (1981), 3.

Rahmlow, Harold F., Fratini, Robert C., and Ghesquiere, James R. PLATO. Englewood Cliffs, NJ: Educational Technology Publications, Inc, 1980.

Ramsey, H. R., Atwood, M. E., and Van Doren, J. R. "Flowchart versus Program Design Languages: An Experimental Comparison." Communications of the ACM 26, 6 (1983), 445-549.

Robertson, G., Mc Cracken, D., Newell, A. "The ZOG Approach To Man-Machine Communication." London, England: Academic Press, Inc., 1981, 461-488.

Robinson, L. "An Orientation to Computer Technology." in Margolin, J. B. and Misch, M. R., Eds., Computers In The Classroom. New York, NY: Spartan Books (1970), 5-61.

Roblyer, M. D. "When Is It "Good Courseware"? Problems In Developing Standards For Microcomputer Courseware." Educational Technology 10 (1981), 47-54.

Roecks, Alan L. "How Many Ways Can The Computer Be Used In Education? A Baker's Dozen." Educational Technology 9 (1981), 16.

Roth, Joel A. "CAI-An Overview." Computerized Educational Technology. 1, Detroit, MI: Management Information Services, 5-12.

Rushby, Nicholas J. Computers In The Teaching Process. New York, NY: Halsted Press, 1979.

Santos, S. M. dos and Millan, M. R. "A System for Teaching Programming by Means of a Brazilian Minicomputer." In O. Lecarmi and R. Lewis (eds.), Computers in Education, IFIP (Part 1). Amsterdam: North-Holland, 1975, 211-216.

Schuyler, J. A. "Programming Languages for Microprocessor Courseware." Educational Technology 19, 10 (1979), 29-35.

Self, John A. "Student Models in Computer-Aided Instruction." International Journal of Man-Machine Studies. 6, (1974), 261-276.

Shapiro, S. C. and Kwasny, S. C. "Interactive Consulting via Natural Language." Communications of the ACM 18, 8

(1975), 459-562.

"Special Section On Machine Learning." SIGART NEWSLETTER 76, Apr (1981), 25.

Silver, Greald A., and Silver, Joan B. Computer Algorithms and Flowcharting. New York, NY: McGraw-Hill, Inc., 1975.

Skinner, B. F. "Why We Need Teaching Machines." Harvard Education Review 31 (1961), 377-398. Reprinted in J. P. De Cecco (ed.), Educational Technology. New York: Holt, Rinehart, and Winston, 1964, 92-112.

Sleeman, D., and Brown, J. S. Intelligent Tutoring Systems. New York, NY: Academic Press Inc, 1982.

Smith, P. R. Computer Assisted Learning. Elmsford, NY: Pergamon Press Inc, 1981.

Smith, R. L., Graves, H., Blaine, L. H., and Marinov, V. G. "Computer-Assisted Axiomatic Mathematics: Informal Rigor." in Computer Education. (O. Lecarne and R. Lewis, Eds.) North Holland, Amsterdam, 1975.

Spitler, C. Douglas, and Corgan, Virginia E. "Rules For Authoring Computer-Assisted Instruction Programs." Educational Technology 11 (1979), 13-20.

Soloway, E., Rubin, E., Woolf, B., Bonar, J., and Johnson, W. L. "MENO-II: An AI-Based Programming Tutor." Journal of Computer-Based Instruction 10, 1 & 2 (1983), 20-34.

Steinberg, Esther R. Teaching Computers To Teach. Hillsdale, NJ: Lawrence Erlbaum Associates, Inc., Publishers, 1984.

Su, S. Y. W. and Eman, A. E. "Teaching Software Systems on a Minicomputer: A CAI Approach." In O. Lecarmi and R. Lewis (eds.), Computers in Education, IFIP (Part 1). Amsterdam: North-Holland, 1975, 223-229.

Sugarman, R. "A Second Chance for Computer-Aided Instruction." IEEE Spectrum 15, 8 (1978), 29-37.

Suppes, P. "On Using Computers to Individualize Instruction." In D. D. Bushnell and D. W. Allen (eds.), The Computer in American Education. New York: John Wiley, 1967, 11-24.

Suppes, P., and Morningstar, M. Computer-Assisted Instruction at Stanford, 1966-68: Data, Models, and Evaluation of the Arithmetic Programs. New York: Academic Press, 1972.

Suppes, P. "Current Trends in Computer-Assisted Instruction." In M. C. Yovits (ed.), Advances in Computers (Vol. 18). New York: Academic Press, 1979, 173-229.

Suppes, P. and Macken, E. " The Historical Path from Research and Development to Operational Use of CAI." Educational Technology 18, 4 (1978), 9-12.

Suppes, P. and Sheehan, J. "CAI Course in Logic." In P. Suppes (ed.), University-Level Computer-Assisted Instruction at Stanford: 1968-1980. Stanford, CA: Institute for Mathematical Studies in the Social Sciences, 1981, 193-225.

Tagg, W. "A Command Language For C.A.I." Computer Education NOV (1981), 29.

Taylor, Robert P. The Computer In The School: Tutor, Tool, Tutee. New York, NY: Teachers College Press, 1980.

Ulloa, Miguel. "Teaching And Learning Computer Programming: A Survey Of Student Problems, Teaching Methods, And Instructional Tools." SIGCSE BULLETIN 12, 2, Jul (1980), 48-64.

Walker, Decker F., and Hess, Robert D. Instructional Software. Belmont, CA: Wadsworth Publishing Company, 1984.

Ward, D. L. and Irby, T. C. "Classroom Presentation of Dynamic Events Using Hypertext." SIGCSE Bulletin 13, 1 (1981), 126-131.

Wardlow, A. "Computer Assisted Learning." Computer Education Feb (1983), 12.

Watson, Paul G. Using The Computer In Education. Englewood Cliffs, NJ: Educational Technology Publications, Inc, 1972.

Wexler, J. D. "Information Networks in Generative Computer-Assisted Instruction." IEEE Transactions on Man-Machine Systems MMS-11, 4 (1970), 181-190.

Wilkes, Sharon J. C. "C.A.L. In Language Teaching At King

Edward VI Five Ways School." Computer Education Feb (1980), 7-11.

Williams, D. M. "C.A.L. Portability and Documentation." Computer Education , 3-5.

Willis, Jerry W., Johnson, D.L., and Dixon, Paul N. Computers, Teaching & Learning. Beaverton, OR: dilithium Press, 1983.

Wirth, Niklaus. Algorithms + Data Structures = Programs. Englewood Cliffs, NJ: Printice-Hall, Inc., 1976.

APPENDIX

SAMPLE ALGORITHM CONTROL FLOW

Affects of the Create/Add Algorithm Module

Provided is a sample demonstration of the create/add algorithm control flow. Each example lists the block numbers annotated at the top right of each flowchart symbol in the chronological order in which they would occur for the scenerio given. Figure 33 serves as a template to assist in the visualization of the actions being performed.

Legend:

X, A	= subject identification variable
Y, B	= topic identification variable
Z	= screen identification variable
C	= answer identification variable

NOTE: All control flow examples originate at the beginning of the algorithm module and precede to the return or stop symbol.

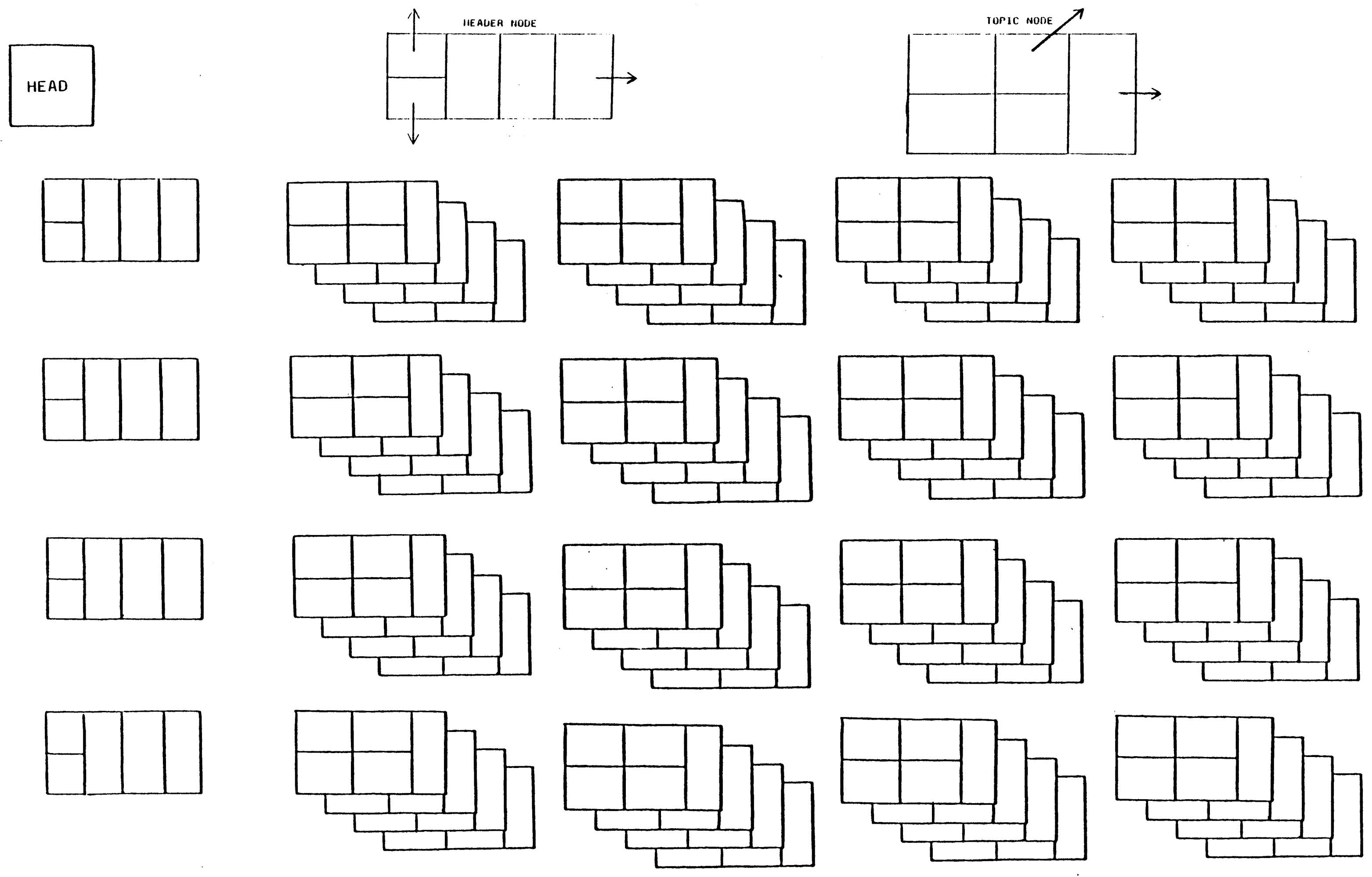


Figure 33. Algorithm Control Flow Template

CREATE/ADD a screen to an empty presentation system. (X=3, Y=3, Z=2)

Blocks: 1, 2, 3, 4, 5, 6, 7, 8, 9, 15, 17, 2, 3, 16.

ADD a second screen to the end of the screen list, same subject and topic as in previous example. (X=3, Y=3, Z=4)

Blocks: 1, 2, 3, 4, 5, 6, 7, 27, 28, 32, 34, 35, 39, 2, 3, 16.

ADD a third screen to the interior of the screen list, same subject and topic as in previous example. (X=3, Y=3, Z=3)

Blocks: 1, 2, 3, 4, 5, 6, 7, 27, 28, 32, 34, 35, 36, 37, 2, 3, 16.

ADD a fourth screen to beginning of screen list, same subject and topic as in previous example. (X=3, Y=3, Z=1)

Blocks: 1, 2, 3, 4, 5, 6, 7, 27, 28, 32, 34, 35, 36, 38, 40, 44, 41, 43, 2, 3, 16.

ADD another screen to the end of the topic list, same subject as in previous example. (X=3, Y=4, Z=1)

Blocks: 1, 2, 3, 4, 5, 6, 7, 27, 28, 29, 31, 2, 3, 16.

ADD another screen to the beginning of the topic list, same subject as in previous example. (X=3, Y=1, Z=1)

Blocks: 1, 2, 3, 4, 5, 6, 7, 27, 28, 29, 30, 2, 3, 16.

ADD another screen to the interior of the topic list, same subject as in previous example. (X=3, Y=2, Z=1)

Blocks: 1, 2, 3, 4, 5, 6, 7, 27, 28, 29, 30, 2, 3, 16.

ADD another screen to the beginning of the subject list, new topic. (X=1, Y=1, Z=1)

Blocks: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 17, 2, 3, 16.

ADD another screen to the interior of the subject list, new topic. (X=2, Y=1, Z=1)

Blocks: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 18, 17, 2, 3, 16.

ADD another screen to the end of the subject list, new topic. (X=4, Y=1, Z=1)

Blocks: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 14, 18, 17, 2, 3, 16.

ADD a screen that already exists, should produce a message to the user. (X=1, Y=1, Z=1)

Blocks: 1, 2, 3, 4, 5, 6, 7, 27, 28, 32, 34, 33, 2, 3, 16.

ADD a new answer to the answer file. (A, B, C=any natural number)

Blocks: 1, 2, 3, 4, 19, 20, 21, 22, 23, 24, 25, 45, 2, 3, 16.

ADD an answer that already exists to the answer file, should produce a message to the user. (A, B, C=any natural number)

1, 2, 3, 4, 19, 20, 21, 22, 26, 2, 3, 16.

Affects of the Delete Algorithm Module

Provides a sample demonstration of the control flow performed by the delete algorithm module. Control flow is illustrated by utilizing the block numbers at the top right of each flowchart symbol in the order of occurrence. Refer to Figure 33 as a tool for understanding the operations being performed.

DELETE the only screen from the presentation system. (X=3, Y=3, Z=1)

Blocks: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 24, 25, 31, 30, 35, 2, 3, 17.

DELETE the interior screen from the sceen list consisting of three (or more) screens in an environment of only one subject and topic list. (X=3, Y=3, Z=2)

Blocks: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 2, 3, 17.

DELETE the last screen from a screen list consisting of two (or more) screens in an environment of only one subject and topic list. (X=3, Y=3, Z=3)

Blocks: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 24, 32, 2, 3, 17.

DELETE the first screen from a screen list consisting of two (or more) screens in an environment of only one subject and topic list.

Blocks: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 2, 3, 17.

DELETE the only screen of a screen list in the second topic list consisting of only one topic in an environment of three (or more) subjects. (X=2, Y=1, Z=1)

Blocks: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 24, 25, 31, 30, 29, 28, 27, 2, 3, 17.

DELETE the only screen of a screen list in the second topic list consisting of two (or more) topics in an environment of three (or more0 subjects. (X=2, Y=1, Z=1)

Blocks: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 24, 25, 26, 2, 3, 17.

DELETE the only screen of a screen list in the first topic list consisting of only one topic in an environment of three (or more) subjects. (X=1, Y=1, Z=1)

Blocks: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 24, 25, 31, 30, 29, 28, 33, 2, 3, 17.

DELETE the only screen of a screen list in the last topic list consisting of only one topic in an environment of three (or more) subjects. (X=3, Y=1, Z=1)

Blocks: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 24, 25, 31, 30, 29, 34, 2, 3, 17.

DELETE the only screen of the screen list from the first, interior, or last topic in the topic list in an environment of two (or more) topics within a single subject. (X=1, Y=1, Z=1)

Blocks: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 24, 25, 26, 2, 3, 17.

DELETE a screen which does not exist, should produce a message to the user. (X=1, Y=10, Z=1)

Blocks: 1, 2, 3, 4, 5, 6, 7, 8, 9, 16, 2, 3, 17.

DELETE an answer that exists on the answer file. (A, B, C=any natural number)

Blocks: 1, 2, 3, 4, 18, 19, 20, 21, 22, 2, 3, 17.

DELETE an answer that does not exist, should produce a message to the user. (A, B, C=any natural number)

Blocks: 1, 2, 3, 4, 18, 19, 20, 21, 23, 2, 3, 17.

VITA

Lee Merle Colaw

Candidate for the Degree of

Master of Science

Thesis: ALGORITHM DESIGN OF A COMPUTER AIDED INSTRUCTION PACKAGE FACILITATING THE INSTRUCTOR-STUDENT RELATIONSHIP

Major Field: Computing and Information Science

Biographical:

Personal Data: Born in Tulsa, Oklahoma, May 1,1953, son of Merle C. and Lois Lee Colaw. Married to Cheryl A. Colaw on September 13, 1975.

Education: Graduated from Will Rogers High School, Tulsa, Oklahoma, in May, 1971; received Bachelor of Science Degree in Zoology from Oklahoma State University in May, 1975; completed requirements for the Master of Science degree at Oklahoma State University, in December, 1985.

Professional Experience: Professional Soldier, US Army, May, 1975 to present; Senior Computer Systems Analyst, 3d Support Command, Frankfurt, Germany, October, 1980 to September, 1981; Commander, 17th Data Processing Unit, 3d Support Command, Frankfurt, Germany, September, 1981 to June, 1982; Executive Officer, 17th Data Processing Unit, 3d Support Command, Frankfurt, Germany, June, 1982 to July, 1983.