INTERACTIVE COMPUTING FOR BLIND CHILDREN

by

James Arthur Woodard, Jr.

A Thesis Submitted to the Faculty of the

DEPARTMENT OF SYSTEMS AND INDUSTRIAL ENGINEERING

In Partial Fulfillment of the Requirements
For the Degree of

MASTER OF SCIENCE
WITH A MAJOR IN SYSTEMS ENGINEERING

In the Graduate College

THE UNIVERSITY OF ARIZONA

1 9 7 2

STATEMENT BY AUTHOR

This thesis has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: James A Woodard Jr

APPROVAL BY THESIS DIRECTOR

This thesis has been approved on the date shown below:

W. R. Ferrell

W. R. Ferrell
Professor of Systems Engineering

May 8, 1972
Date

ACKNOWLEDGMENTS

The author wishes to thank Dr. William R. Ferrell for his expert guidance throughout all phases of this project, as well as Dr. Roger J. Weldon and Dr. Robert L. Baker for their comments in regard to the preparation and content of this thesis. Many thanks are also given to the faculty of the Arizona School for the Deaf and Blind (ASDB), without whose patience and cooperation the project could not have been accomplished. Finally, special appreciation is expressed to the children from ASDB who willingly put their time and effort into learning to program the computer; and to Joyce, who willingly put her time and effort into typing, editing and critiquing this paper.

TABLE OF CONTENTS

TABLE OF CONTENTS--Continued

TABLE OF CONTENTS--Continued

LIST OF TABLES

LIST OF ILLUSTRATIONS

ABSTRACT

This project consisted of teaching visually handicapped children to program a braille-adapted PDP-8/e computer, with particular emphasis on mathematical applications. Most visually handicapped children are slower in learning mathematics than their sighted contemporaries. It was hypothesized that an interactive computer could be an invaluable asset by providing a responsive environment capable of relieving them of the burden of calculation, providing them with a tool for numerical exploration, reinforcing their basic number concepts and motivating them. The objective was to identify the problems related to such a usage of computers and their introduction to very young students.

The findings of the project were that elementary level blind students could learn to use the computer as a calculation aid while eighth grade students could learn general programming techniques. The computer lessons motivated the students and increased their self-confidence; however, the limited nature of this project necessitated subjective evaluation methods. Before a detailed project could be conducted, a more reliable method of measuring student progress should be developed. Finally, good quality braille can be produced on a PDP-8/e computer with some inexpensive modifications.

CHAPTER 1

LOW MATHEMATICAL ACHIEVEMENT IN VISUALLY HANDICAPPED CHILDREN

Introduction

The problem of low mathematical achievement by blind children in comparison to their sighted peers has been recognized for more than thirty years. Recently, computers have appeared to possess the characteristics necessary to alleviate some of the proposed causes of this low achievement in mathematics. This project was an attempt to answer such related questions as: can young blind students learn to use a computer; and, what are the difficulties the young students will encounter in computer instruction? Consequently, the project consisted of teaching computer usage to young blind students and evaluating, necessarily subjectively, their progress.

Background

Hayes (1944) discovered a 20 percent retardation in mathematics while evaluating the Stanford Achievement Tests for use with blind children. For that reason, he concluded that the Stanford Achievement Tests were inappropriate for testing the visually handicapped. Later evaluations of the same tests by Nolan and Ashcroft (1959) led to a similar estimate of a 16 percent retardation but they found the tests were sufficiently reliable for practical use, provided that the time limit of the tests be increased for the visually impaired so they would have an

opportunity to attempt the same number of problems as the sighted children.

Poor Basic Number Concepts

Riddle (1948, p. 86) identified what he termed "the most retarding factors in learning seventh and eighth grade arithmetic." His first and most emphasized factor was that the children lacked basic number concepts. Counting tended to be an exercise in rote memory, rather than in rational thinking. Riddle concluded that it was impossible to make any meaningful progress in arithmetic if the child did not understand numbers. He recommended exercises in counting objects that were interesting to count, combined with a limited amount of abstract counting. Riddle identified other retarding factors, e.g., mixing the four fundamental arithmetical processes too soon, starting fractions before the fundamental processes are understood, etc; however, he subordinated all of them to a lack of basic number concepts. It appears reasonable that if these retarding factors were as prevalent as he claimed, the retardation among visually handicapped children may well have been higher than 20 percent.

The same lack of number concepts was identified by Smith (1964, p. 50) as one reason "that arithmetic is one of the most difficult subjects for blind children to master." The subject of his investigation was the Catherine Sterns Materials adapted for teaching mathematics to blind children. In discussing the use of these materials at the Perkins School for the Blind he characterized them as, "...the best

materials we have found to date to bring understanding of basic number concepts to our very young blind children." He supports Riddle's argument for more applied counting by stating that the understanding comes through the child's participation in the various counting experiences that the materials are designed to accomplish.

A survey of residential schools and resource rooms for the blind was conducted by Lewis (1970, p. 70) to discover which of the various calculation aids available were the most preferred. One question asked was: "How do the braille students compare on standard test scores with the sighted test norms?" Eighteen of the 36 respondents to this question stated that the braille students were lower, although they did not say how much lower. The most frequently listed problem was that the blind student's "...limited contact with his environment causes a delay in his understanding of basic concepts of numbers in comparison with sighted peers."

Computational Difficulty

A second possible cause of retardation is computation difficulty. The inability to do computations readily would seem to make it more costly in terms of effort and time for the blind to use math and to experiment with mathematical ideas. Many computation aids are available, ranging from a modified abacus to a type slate and braille writer. The difficulty is that the aids have been designed to meet primarily tactile requirements and, as a result, possess few of the mathematical characteristics they attempt to portray. For example, working an arithmetic problem of moderate difficulty on either of the

previously mentioned aids is largely a mental process in which the aid functions only as a memory aid. The student is required to understand the logical sequence of mathematical operations, he must remember the rules governing the operation of the computation aid, and do all of the calculations mentally. Remembering the rules and possessing the ability to do mental calculations are of secondary importance to knowing the logical sequence of operations: in fact, they are almost unrelated to mathematical understanding. The aid merely keeps proper place values. Although some may argue that that is the sole purpose of a computation aid, it also helps perpetuate the retardation. The student is required to spend an inordinate amount of time in repetitious and mathematically unrelated activities just to maintain place values. He performs all of the mathematical operations mentally, without any aid. Mathematics quickly becomes boring, error ridden and frustrating. This may be another reason why he is retarded at mathematics when compared to his sighted peers.

Lack of Visual Representation

The third proposed cause of the retardation is the inability of blind children to use visual representations of numerical and mathematical concepts to aid thought. The sighted individual can use pencil and paper to represent and explore mathematical concepts, determine consistencies or inconsistencies and discover solutions. Some calculation aids may help the blind in this area, e.g., a geometry aid, but they are necessarily specialized and not universally

applicable. Consequently the blind student does not have an external perceptual environment in which mathematical operations, rules, results, etc. can readily be represented and which can serve as a stimulus to creative thought.

Poor Interest and Motivation

A final probable cause of retardation is the feedback effect on motivation and interest of the blind student's early failures and continuing lack of mastery of mathematics. Each failure is related to the previous one and thereby may reinforce the student's feelings of inadequacy or lack of interest.

Other Influencing Factors

Other factors which may influence the visually handicapped student's progress in mathematics are included here to illustrate the multiple dimensions of the problem. The current changeover to "new math" and to new codes for brailled mathematics are two of these (Lewis 1970). Emphasis on mental arithmetic has generally been recognized as another culprit (Nolan 1967) and consequently its place in current curricula has been diminished.

Identification of the probable causes of retardation is the first step toward finding a solution. In this case, the interrelationships of the proposed causes and the recent advent of an inexpensive method of producing braille on a teletype suggests that a computer might help solve the problem. That possibility was the basis for this project.

CHAPTER 2

HYPOTHESES CONCERNING THE POTENTIAL EFFECT ON MATHEMATICAL RETARDATION OF TEACHING COMPUTER PROGRAMMING SKILLS TO THE VISUALLY HANDICAPPED

A computation aid for the visually handicapped should combine mathematical relevancy with an ability to interact with the student. It should be able to help with all kinds of problems, require a minimum of routine work by the student and indicate when erroneous operations are attempted. The device would have to meet the tactile requirements of the blind, just as the other aids do, but optimally it should go beyond being a place keeper to remove or reduce as many of the hypothesized causes of retardation as possible. Based on the following hypotheses, computers appear to possess these characteristics.

Basic Number Concepts Hypothesis

The computer can be used as a calculator to aid the blind student in improving his basic number concepts.

Justification:

A computer that has been preprogrammed to accept the fundamental mathematical operations, i.e., addition, subtraction, multiplication and division, has the capacity to work problems of practically any degree of difficulty, quickly and efficiently. This will be referred to as the "calculator" mode of operation, as opposed to the "programmed"

mode. A calculating machine combines high standards of correctness with a tireless source of answers to readily explain to its user the consequences of number operations. In comparison to the normal calculating machine, the computer possesses the same qualities; however, it is much simpler to use, faster and the way it must be instructed more nearly reflects true mathematical logic. Under the control of a young blind student, the computer calculator could perform numerous computations in a minimum amount of time and thereby help reinforce number concepts at an early age -- provided a young blind person could learn to use it.

Computational Difficulty Hypothesis

In either the calculator mode or the "programmed" mode a computer can relieve the blind student's computational difficulties.

Justification:

Concurrent with developing his number concepts using the calculator mode, the blind student could be taught general purpose programming of the computer. This would enable him to work increasingly difficult problems without the burden of calculation. This does not mean that the student could work a problem that he did not understand or could not set up in a logical sequence. Being relieved of the burden of calculation is not the same as being relieved of the necessity for logical thought. On the contrary, it should encourage more cognitive activity toward the solution of the problem.

Lack of Visual Representation Hypothesis

The computer can provide an environment for the blind child to use for discovering mathematical principles.

Justification:

The increased perceptual activity, brought about by relieving the burden of calculation, might also stimulate the student to use the computer for exploration. By so doing, the computer becomes the "pencil and paper" for the blind student. It provides a testing ground for reliable experimentation toward finding the same consistencies or inconsistencies, testing hunches, etc., analogously to the way the sighted child does with pencil and paper. For example, if a sighted child were told that the area of a one-by-one square was the same as the sum of the areas of two right triangles with one-by-one sides, it would be a simple task for him to examine the veracity of the statement using graphic means. Obviously, the blind child would be at a disadvantage if he were given the same problem. He certainly could calculate the areas of the figures and compare their totals; however, the effort he would have to expend would tend to discourage him. On the other hand, using the computer he could not only quickly verify the original problem but could experiment to find other combinations of triangles that met the same criteria. This could be valuable time spent in enthusiastic experimentation and learning, time that might otherwise be lost.

Interest and Motivation Hypothesis

Learning to use a computer can increase the student's interest and motivation toward mathematics.

Justification:

The final hypothesized benefit of the computer is the fact that since visually handicapped children have never used a computer, they have never experienced failure on it. Therefore, previous failures in mathematics are less likely to affect their motivation for learning to program. The stimulation of a new experience, subsequent mastery of the computer and the cumulative reinforcing effect of successes along the way should combine to motivate the student even more. The step-by-step transfer of programming principles to mathematical applications should also carry with it motivation to help allay the student's feelings of inadequacy and disinterest in mathematics. This is based on the student's eventual realization that he has the means to cope with his mathematical difficulties.

CHAPTER 3

OVERVIEW OF THE PROJECT

The project consisted of teaching visually handicapped children to program a braille-adapted PDP-8/e computer. The children were all elementary and junior high students from the Arizona School for the Deaf and Blind (ASDB). The primary emphasis of the project was to teach the children the mathematical applications of the programming techniques.

Project Objectives

The overall objective must be to determine whether or not a knowledge of computer programming techniques can affect the student's mathematical achievement. Inherent in this objective is a determination of the feasibility of teaching programming to visually handicapped students; and the optimum grade level to begin such instruction. This is a relatively new approach at attempting to alleviate the causes of the mathematical retardation of the blind. The Perkins School for the Blind has the only other project known to the writer in which programming is taught to elementary level children.

Secondary objectives consisted of testing the means for obtaining readable braille output from the computer. If this modification proved to be a satisfactory method of producing "readable" braille then there would be other beneficial uses for it, e.g., document reproduction, etc.

Another secondary objective was to furnish management related information to ASDB. The ASDB had been asked to project uses it might have for a state wide time-shared computer system. Additionally, they have instituted data processing training for their deaf students and would like to know the feasibility of making similar arrangements for the blind students.

General Operation of the Project

Planning Constraints

As soon as planning for the project began it became evident that resource constraints would not permit an in depth statistical evaluation of the effects of computers on the mathematical retardation of visually handicapped children. There were a limited number of braille students available from ASDB. A braille student is one whose degree of blindness requires that he work in braille. Programming lessons had to be scheduled with variable period lengths so that they would interfere as little as possible with the normal school activities, of which there were many. The classes had to be held at The University of Arizona, since the computer was located there; transportation of students would normally be one at a time because the instruction was to be on an individual basis. Non-braille projects on the same computer also limited the amount of time instruction could be given each day. Based on these constraints, it was decided that a small study would be sufficient to test the feasibility of teaching computer programming to young blind students, and to assess in a general way whether the

hypotheses about the advantages of computer usage might be correct. It would also attempt to satisfy the secondary project objectives to some degree.

Language Requirement

The need for an easy-to-learn computer language was evident. There was no desire to introduce more learning hardships by using a difficult computer language. The project required the language to have a calculating mode, i.e., the four fundamental operations, and a programming mode, i.e., capable of more involved procedures. Optimally there should be as few usage differences as possible between the two modes, in order to maximize the transfer of principles from calculating to programming.

The computer language also had to be capable of reciprocating with the student to the extent of following instructions, replying to inquiries, pointing out errors, etc. Such languages are termed "interactive" and as such, possess various internally defined instructions that the computer recognizes and follows according to established rules. Many interactive language commands parallel the language of the operator, e.g., GO, TYPE, QUIT, etc. and hence are easier to learn.

Outline of the Project

Eight students were chosen to participate in the project. They were selected as low achievers in mathematics, as rated by their own teachers. Before any classes could take place, lessons had to be written, the braille modifications made to the computer and the logistical

requirements solved. Shortly after the completion of these tasks, the students were taught to use the computer as a calculator, and finally taught how to program it to solve more difficult problems. Each student's progress was rated during the lessons by the programming instructor. At the conclusion of the lessons the student's teacher rated the effects of the programming instruction on his mathematical abilities, attitudes, etc. Additionally, the students were given the opportunity to comment on the lessons.

The specific instruction given and its results will be discussed in subsequent chapters.

CHAPTER 4

DESCRIPTION OF COMPUTER MODIFICATIONS

The PDP-8/e computer consists of a teletype console and a combination data processor/memory unit. Both hardware (mechanical) and software (program) changes were necessary to enable the computer to type in braille. The hardware changes were all made to the teletype console while the software modifications consisted of essential changes to the programming language.

General Hardware Modifications

The teletype machine, Teletype Corporation Model 33, was the only device used by the subjects for input/output with the PDP-8/e computer. It was modified in such a manner as to enable the subjects to perform all of the functions a sighted individual would be able to perform. Each modification is briefly discussed below and then certain of the more involved ones are described in greater detail in Appendix A.

Print Head

The print head was modified by removing the letters from a normal print head and substituting steel pins in all the various possible combinations that can make up one-half of a braille cell (Anderson and Rogers 1968). See Figure 1 and Appendix A. During operation, the

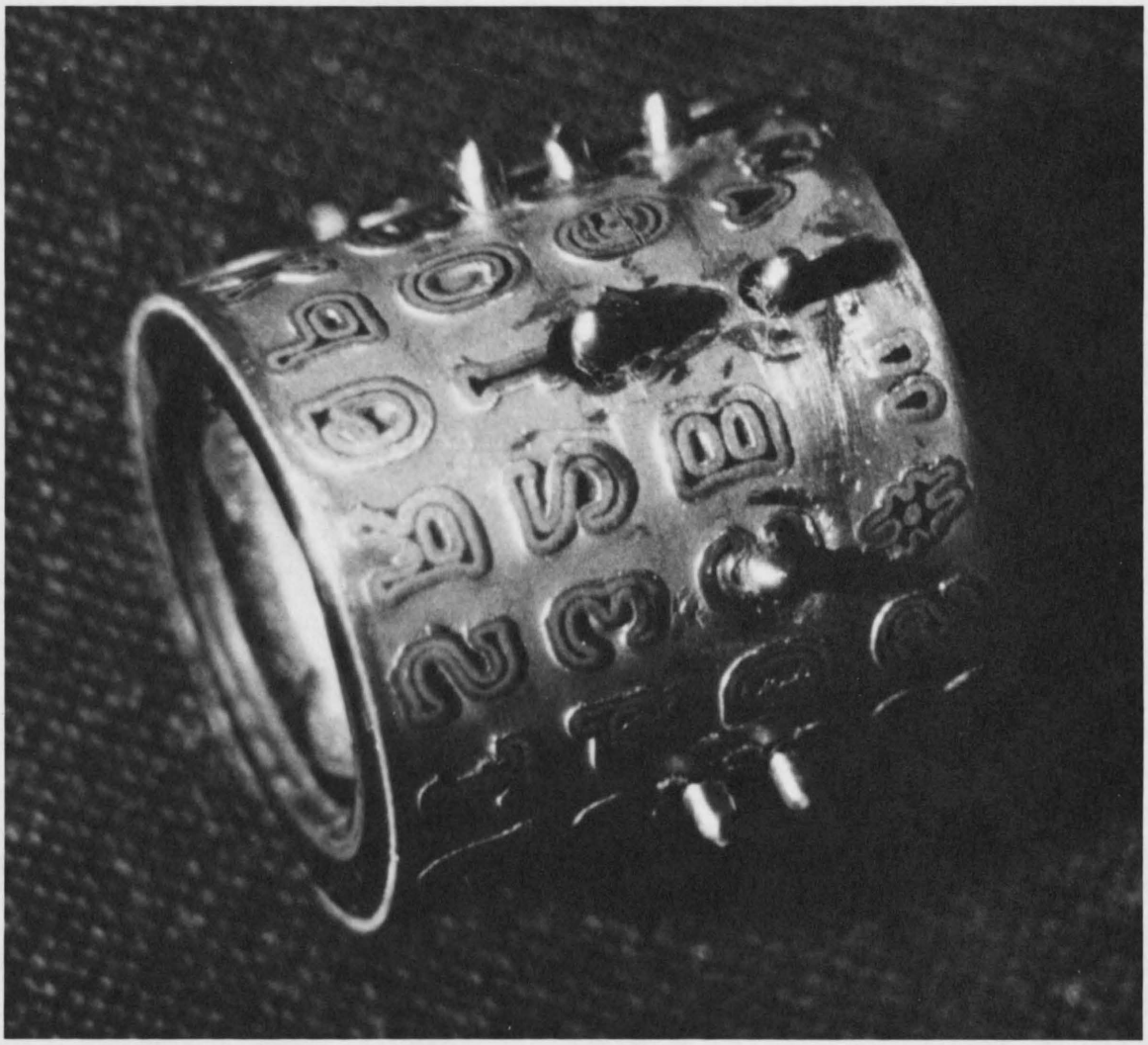


Figure 1. Modified Print Head.

braille is embossed upside-down as the operator faces it. Since the print head embosses into the paper, the characters must be read from the reverse side. To read the paper the operator folds it down toward himself, which turns the braille rightside-up.

Platen

Since it is necessary for the print head to emboss the paper rather than punch holes in it, the platen was softened to cushion the strike of the print head against the paper. The hard rubber covering was removed from a spare platen and then it was recovered with 1/16 inch surgical tubing. This is a permanent modification, so two platens are used, one for braille operation and one for print. See Appendix A for more details.

Keyboard

The necessity of this modification depends largely on the typing ability and keyboard knowledge of the individual using the teletype. In this case it was necessary to braille the entire keyboard with the exception of five keys, i.e., the "Here Is" key, the "Line Feed" key, the "Repeat" key, the "Alternate Mode" key, and the "Break" key, none of which were required in this application of FOCAL programming. The braille for the keyboard was produced on a Dymo Tapewriter that embosses braille. The characters were all an abbreviated form of braille, known as teletype braille, which uses only one cell per character, as shown in Figure 2. The tapewriter normally embosses only the alphabet, but with improvisation it can be used for the additional symbols. There is a clear

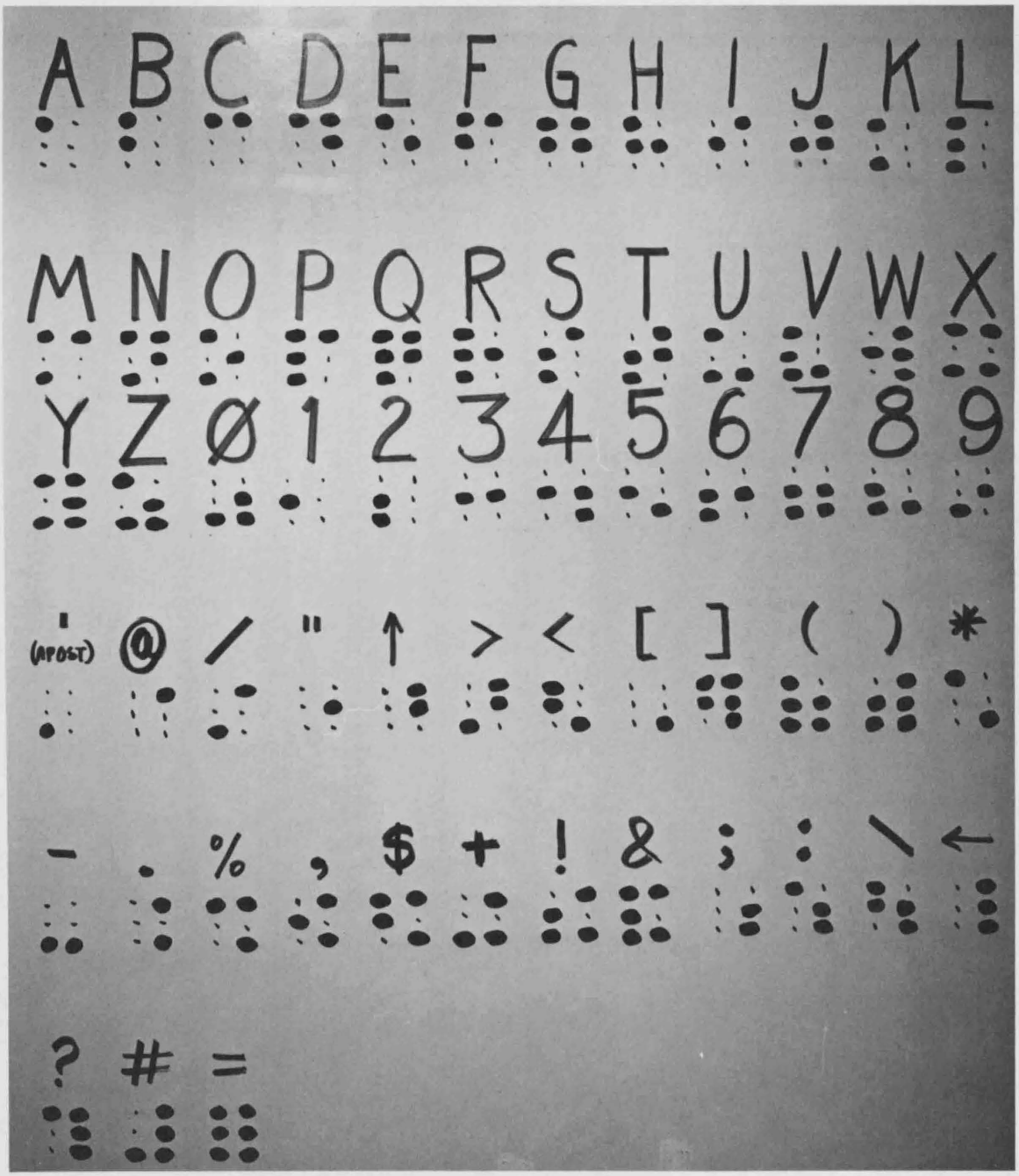


Figure 2. Teletype Braille Characters.

Source: Final Report to SRA, 1970.

plastic tape available for the tapewriter that permits the print letters to be seen through the braille, thereby allowing the braille to remain on the keyboard permanently. See Appendix A.

Window

The original window on the teletype was removed and a new one was made from 1/16 inch thick plexiglass. This is not a required modification; however, with it fewer line feeds are needed to bring the braille line up to the point where it can be read and it consequently saves paper. A similar modification was described in the Anderson and Rogers article (1968).

Paper

Standard poster (newsprint) or bond teletype paper does not have sufficient weight to allow proper embossing; the bumps are not durable. Adequate paper was found to be 20 pound (sulphite) General Copy Bond for an IBM Copier. It is distributed by Graham Paper Company of St. Louis, Missouri. It comes on rolls that are 8 1/2 inches wide by 6 3/4 inches diameter with a 3 inch cone. The rolls are too heavy and thick to fit in the normal paper position on the teletype. Therefore, a small wooden stand was built to hold the roll and placed behind the teletype. Another possible solution to this problem would be to rewind some of the IBM Copier paper onto an old teletype paper spool, and then place it in the normal paper position.

General Software Description

The PDP-8 series of computers use an interactive language called "FOCAL," an acronym for FOrmula CALculator. FOCAL was advantageous for this project because it had the necessary allowances for the different ability levels of the subjects. In other words, it would allow the student to go from simple on-line computing, i.e., addition, subtraction, etc., to more difficult programming techniques, e.g., compounding interest, computing areas, making decisions based on results, etc. It also had subprograms available, called "FUNCTIONS," which allow computation of various often used mathematical operations, e.g., square roots, sines, exponentials, etc., although these were not used in this project. The more FUNCTIONS, or subprograms, a language has the less room there is in the computer's memory for performing calculations. Another advantage of FOCAL was that some of the unneeded FUNCTIONS could be removed to allow room for the essential modifications to make the computer print in braille. Hence, the braille modifications to the software did not significantly affect the size of the calculation area in the computer's memory. See Appendix B.

Braille Subprogram

When either the calculating or programming mode of FOCAL is used, the teletype printer has two methods of operating. As long as the student uses the correct procedure, the teletype prints the character corresponding to each key he pushes. This allows the student to monitor his progress. When he uses an incorrect procedure the teletype

either indicates the error immediately or during its attempt to follow the incorrect instruction, depending on the nature of the error. This illustrates an important characteristic of FOCAL. The computer, rather than the keyboard, controls the print head. It is this characteristic which enables the insertion of a subprogram to automatically change everything to the proper code for the braille print head. Therefore, the entire process is printed in braille.

As previously mentioned, the braille print head prints one-half of a braille cell at one time. Consequently, each braille cell uses three normal printer spaces, one for each half cell and one space to adequately separate each of the cells. Therefore, the maximum number of braille cells possible on the normal 72 space line is 24. A listing of the braille subroutine written by Steven Silberstang and the author is provided in Appendix C.

Line Feed Subprogram

This is a small subprogram inserted into the FOCAL program to place seven line feeds (vertical spaces) between each line of braille. This allows the user to read each line as soon as the braille print head goes to the next line. See Appendix D.

Braille Subprogram Problems

There are two minor problems in the braille subprogram. One problem involves the 24 braille characters per line and the other concerns two minor characters that are not printed in braille. Both problems are discussed in detail in Appendix B.

CHAPTER 5

DESCRIPTION OF THE LESSONS

Five lessons were planned to give the students a basic knowledge of the FOCAL language and its applications. Each lesson emphasized the flexibility and consequent potential of the computer and encouraged the student to harness its power for his own use.

Planning Criteria

The lessons were designed to incorporate the four hypotheses which related computers to improving the mathematical achievement of the visually handicapped. Although the hypotheses could not be tested directly, a determination of the children's programming potential would be an indication of the appropriateness of the hypotheses. Were the children too old or too young? Did the emotional maturity have any effect on their performance? Does poor mathematical ability cause the student to be at a disadvantage during computer lessons? Can the children muster sufficient motivation to complete the lessons? These are some questions the project sought to answer. The possible discovery of previously unrecognized questions was also considered in the planning. The answers to these questions held a key to the validity of the hypotheses, even though they would not prove or disprove the hypotheses.

Lesson Attributes

Each lesson incorporated the following characteristics.

Demonstrate the power of the computer. The single aspect of the computer that was expected to interest the student the most was its ability to do all of the mathematical trivia that are normally required of the student. This was expected to be a major consideration in the student's motivation to continue learning to program.

Demonstrate a need before introducing a computer command. By pointing out the need for a particular type of operation, the use of a command is justified to the student beforehand.

Demonstrate applications of principles with exercises. Every new principle was accompanied by an exercise which applied it. This allowed the student to use each principle as he learned it to reinforce that learning. It also was expected to build his confidence in his ability to program. Satisfactory performance on this exercise was used as the criterion for accomplishment of that portion of the lesson.

Reinforce success and encourage experimentation. Each student received praise when he performed an exercise in the prescribed manner. If he were unsuccessful, then that fact was pointed out to him, but without emphasis. The making of mistakes was not discouraged because mistakes on the computer are easy to correct. Mistakes indicated that the student's confidence was building at least to the point where he was not afraid to do something on his own. This was a difficult step for many of the students.

Conduct of review. Starting with the second lesson, a review period was conducted at the beginning of each class. The length of the review period depended on the student's level of achievement on the lessons presented to date. The review period also helped the instructor determine the progress of the student. It was considered an integral part of the lesson and the amount of time spent in review limited the amount of time that was available to cover new material.

Lesson synopses. At the end of each lesson the student was given his own braille synopsis to take home. It reviewed the entire lesson. He was encouraged to read the synopsis at least once as preparation for his next lesson. See Appendix E.

Organization of Lessons

The first two lessons concerned the use of the computer as a calculator, and were the easiest. Both began with the computer's embossing a preprogrammed message designed to interest the student in the lesson and to suggest the existence of non-mathematical uses of computers. If the student were hesitant about learning programming because of a dislike for mathematics, recognition of other applications should have helped to remove his hesitancy. This idea was not exploited, but its implications are grounds for more experimentation. The introductory messages are in Appendix F. The last three lessons were on "general purpose" programming of the computer, with practical applications of the programming techniques to areas of mathematics.

Lesson One: The Four Fundamental Operations

After the introductory message, the first lesson covered how the keyboard of the teletype was arranged, the use of the extra keys, e.g., "Return" and Rubout" keys, and the locations of the four fundamental mathematical operations signs, i.e., addition, subtraction, multiplication and division. All of this was preparation for introducing the TYPE command, which controls the operation of the printer. Once the TYPE command had been taught, the student got his first opportunity to operate the computer by working the sample problems on his lesson synopsis. See Appendix E. It was important to let the student try it immediately so that he would discover that using the computer was as easy as he had been told.

Lesson Two: Priority of
Mathematical Operations

The second lesson concluded the instruction dealing with the computer as a calculator. It began with an introductory message and then introduced parentheses by means of an example of the priority of mathematical operations. In other words, it demonstrated that in the absence of instructions to the contrary, the computer will multiply, divide, add and subtract, in that order; even though the student did not necessarily intend for it to work that way. For example, note the two different answers to the same problem written with and without parentheses in FOCAL format:

```
TYPE 2+2*4
=10
TYPE (2+2)*4
=16
```

The student then understood the need for parentheses and was ready to learn how to use them.

Midway through the second lesson an exercise (see Appendix K) in the use of parentheses required the student to use what he had learned in somewhat more difficult problems. Here he was encouraged to think on his own, to apply principles and to discover that he could do involved calculations on the computer. It was expected that such a discovery would be highly motivating.

At the end of the second lesson the student was introduced to the use of quotation marks for reproduction of text strings as in written messages, and encouraged to use them on his own.

Lesson Three: Line Numbers Combined with the GO and ERASE Commands

The third lesson began the instruction on general purpose programming. Before any actual programming could be done it was necessary for the student to understand line numbers, their sequential relationships, and how they are used. After this portion of the lesson was completed, there was an exercise (see Appendix K) that combined line numbers with the TYPE command. This demonstrated the interrelationships of programming, a fact that became more evident as the student progressed, and gave an indication of the student's learning. Now that the student had line numbers and the TYPE command combined he had to realize that he needed something to make the program operate. At this time the GO command was introduced, along with its related form, the GOTO command.

Now he had everything he needed to program the computer, and he was encouraged to do so.

As soon as the student was ready for the second exercise, he discovered that he did not know how to remove the previous program from the memory. This brought about the introduction of the ERASE command and its alternate forms. The end of the third lesson consisted of a final exercise which required the student to apply all of the principles he had learned throughout the three lessons.

Lesson Four: SET, DO, FOR and QUIT Commands

This lesson began with control characters and covered four additional commands. The control characters consisted of the exclamation mark, used to make the printer go to a new line, and the percent sign, which changes the format between integers and real numbers. There are other control characters available in the FOCAL language, but they were less likely to be used and were omitted. The braille handout for the end of the lesson covered the use of control characters in detail, so relatively little time was spent using them by themselves, but they were used in the applications.

The first command of lesson four was the SET command. The purpose of this command was described in terms of the necessity to identify numbers with names. The importance of this became obvious after the next two commands had been presented.

The exercise for this portion of the class was incorporated into the synopsis for this lesson and it consisted of "naming" numbers with letters and then doing various mathematical operations with the letters, e.g., A+B, C x D, etc. The second command in the lesson was the DO command. It was discussed as an extension of the GO command, for use when a small portion of a program was desired. Finally, the FOR and QUIT commands were jointly introduced as a method of putting all of the commands learned to date into a usable program. When originally planned, the students were to calculate the areas of three similar right triangles by combining the above commands. However, the concept of area was not familiar to most of the students so instead they wrote a program to generate the multiplication table for nine. For example, in FOCAL format:

```
2.1 FOR B=1,1,9; DO 3.0
2.2 QUIT

3.1 SET A=9*B
3.2 TYPE A,!
```

This tells the computer to start with B=1 and add one to it until it gets to B=9. Each time it adds one it multiplies that value times nine and types out the answer. The exclamation mark makes it type each answer on a different line. Following that exercise they were told to modify the program so that it would type out the seven, eight and nine multiplication tables. This was to demonstrate the flexibility of the commands and it did not require any major alterations to the previous programs. One solution was as follows:

```
1.1 FOR C-7,1,9; DO 2.0
1.2 QUIT
```

```
2.1 FOR B=1,1,9; DO 3.0

3.1 SET A=C*B
3.2 TYPE A,!
```

This time C starts with the value seven and B goes through the entire cycle from one to nine. Then C has one added to it and B goes through the cycle again. The third time through C becomes nine and B goes from one to nine for the final time. It may not be obvious to the casual reader, but the SET and TYPE commands could be replaced with "TYPE C*B,!". The SET command was used here simply to give the student practice in applying it. The synopsis for this lesson also contained a detailed coverage of programs similar to those written in class. See Appendix E.

Lesson Five: ASK and IF Commands

Up to now the students had written programs that ran without any interaction or inputs from the programmer once they were started. They were also entirely sequential in their operation. These two limitations were used as justification for the two commands introduced in the fifth lesson.

The ASK command was first used in a program that would ask for any two numbers and then type their product. For example:

```
1.1 TYPE "WHAT ARE YOUR NUMBERS?,"!
1.2 ASK A
1.3 ASK B
1.4 TYPE A*B,!
1.5 GOTO 1.1
```

When this program is started the computer types, "WHAT ARE YOUR NUMBERS?" and waits for two numbers to be typed on the keyboard. Once this has been done by the operator the computer then types their product, goes to a new line and begins all over again.

Next the student was required to change the program so that it would type the sum of the numbers instead. This is a simple change, but one that tested the student's ability to think logically.

The IF command demonstration program was originally designed to complement the triangle area calculator intended to be used in lesson four. It was a program that would tell the operator which of the three triangles he had selected after he typed the length of the base. See Appendix E. Since the first triangle program was not used, this second program was not appropriate. An alternate approach was taken and the student was told to write a program that would ask for a number. If the number were larger than ten, the program would subtract seven from it; if the number were smaller than ten, it would add seven; and if the number were ten, it would simply type it. The following is a possible program:

```
1.1 TYPE "WHAT IS YOUR NUMBER?,"!
1.2 ASK A
1.3 IF (A-10)1.4,1.6,1.8
1.4 TYPE A+7
1.5 GOTO 1.1
1.6 TYPE A
1.7 GOTO 1.1
1.8 TYPE A-7
1.9 GOTO 1.1
```

First the program asks for a number in the same way the preceding program did. It then subtracts ten from the number and if the result is negative the number was less than ten. It would therefore go to line 1.4, add seven to the original number and type out the result. It then starts all over again. The process is similar when subtracting ten yields either a result of zero (when the original number is ten) or a positive result (when the original number is larger than ten).

The final topic in this lesson was looping. It was used in both of the above programs. Once the programs started there was no way out. It was emphasized that such loops should be avoided unless that kind of operation was desired. Options to looping, such as using the QUIT command, were discussed. It was also stated that once the loop had started the only way it could be stopped was from the computer control panel. This was the conclusion of the planned lessons.

Subject Selection and Evaluation

A total of eight braille students participated in the study. All of the subjects were residential students at the Arizona School for the Deaf and Blind (ASDB). They were selected by their teachers based on their demonstrated lack of achievement in mathematics. Selection on this basis was justified by the reasoning that if the low achievers could benefit from learning computer programming, then presumably the other members of the class should do at least as well. Benefits from learning programming should also be more noticeable in these students.

The grade, sex and vision distributions of the subjects are as listed in Table 1.

Special education is a class for students with emotional problems, and P.M. was at about third grade level. The choice of subjects was largely determined by the prerequisite that they be braille students. As previously mentioned, the relatively small number of students from which to choose subjects was a major constraint.

Table 1. Grade, Sex, Age and Vision Distribution of Subjects

Initials	Grade	Sex	Age	Degree of Vision
P.M.	Special Education	M	11	No light perception
I.G.	4	M	11	Light perception both eyes
E.S.	5	M	13	20/400 both eyes
O.S.	5	M	12	Can detect hand movement
B.A.	6	F	12	20/800 left 20/400 right
B.M.	8	F	14	5/300 left, light perception right
M.B.	8	F	15	No light perception
S.C.	8	F	16	Light perception left, none right

Evaluation Methods

Pre-Instruction Evaluation by Teachers. Before the subjects had any contact with the programming instructor, their current mathematical achievement was appraised by the individual teachers. (See Appendix G for a sample evaluation form.) The purpose of the evaluation was threefold. First, it asked for a comparison of each student's mathematical achievement and aptitude to his achievement and aptitude in all other academic areas. Second, it asked the teacher to estimate the student's overall social adjustment compared to all other blind students. And finally, the questionnaire asked for a prediction of the student's future ability to successfully hold a job. Obviously, this questionnaire requires subjective estimates since it deals with

opinions rather than facts. However, lacking more objective measures, the opinions of teachers who know the students well were considered the most reliable indication of the students' performance and potential.

Evaluation of the Students During Instruction. During each class period the student's progress on the applicable lesson was recorded (see Appendix H for a sample evaluation form) to facilitate a later determination of the appropriate grade levels for introducing the various types of programming, and to indicate the students' learning characteristics. This evaluation was the only method used to measure students' progress. The problem of devising a reliable and objective test for assessing students' progress was not solved. It would have to permit comparison of younger and older students, not be subject to bias due to the students' fear of errors, and not introduce anxiety into the learning situation. As a result, it was decided to rely on informed, yet inherently subjective, evaluations by the instructor to determine student achievement. For example, students were required to apply the commands, as they learned them, to problems. Since the lessons were cumulative, previously learned material also had to be applied to each problem. Whether the student could apply the commands without help was one criterion for progress. Other indices were: the amount of help a student needed; the amount of review necessary at the beginning of a lesson; whether the student understood the step-by-step operation of commands well enough to sequence them himself; and

whether the student used the proper format for each of the commands he had been taught. These indicated the students' progress through the lessons.

Teachers' Post-Instruction Evaluation. The teachers were asked whether they noted any effects on the students of their being taught to program a computer. See Appendix I. There were several disadvantges with which the teachers had to contend in order to make such an evaluation. 1) It was hard for them to try to evaluate the effects of programming when they did not have any knowledge of what it encompassed. They were purposely not told what was taking place in order to avoid any unnecessary bias in the experiment. 2) The lessons took place during a period of three and one-half weeks. This amount of time is really not sufficient for a teacher to notice what may be only subtle changes in a student's attitudes or abilities. In fact, the time appears to have been just long enough to give indications of possible changes that would result if the program were to continue. Nevertheless, the teachers were cooperative and provided a basis for some conclusions to be made, although their observations are subject to bias.

CHAPTER 6

RESULTS OF THE PROJECT

The results of each evaluation and lesson will be discussed separately and summarized in a table.

Teachers' Pre-Instruction Evaluation

As previously stated, the children were selected for this project because they were low achievers in mathematics. The pre-instruction evaluation gave an indication of the standing of the student in relation to his visually handicapped peers and was intended to provide a standard by which to judge his progress after the lessons. Table 2 summarizes the results of this evaluation. The questions on the evaluation form are considered in order below.

Present Level of Mathematics Achievement

All of the children were rated at the "below average" mark on the graph, with the exception of one of the fifth grade boys who was rated somewhat higher than that. This is a clear indication that these students were aptly chosen as low achievers.

The reasons the teachers gave for the poor achievement were mainly lack of emotional development, slow computation methods, poor understanding of basic facts, excessive absenteeism, poor work habits and lack of reasoning ability. These reasons support three of the

Table 2. Teachers' Pre-Instruction and Post-Instruction Evaluation

		Pre-Instruction Portion						Post-Instruction Portion
Initials	Grade	Present Level of Mathematical Achievement and Reasons	Present Level of Non-Mathematical Achievement and Reasons	Mathematical Learning Ability	Non-Mathematical Learning Ability	General Adjustment Level	Chance for Successful Employment	
PM	Special Education	Below average - due to emotional problems	Average - no reason given	Poor	Slightly less than average	Poor	25%	Emotional benefit only
IC	4th	Below average - no reason given	Slightly less than average - attitude	Poor	Slightly less than average	Poor	25%	No change noticed
ES	5th	Below average - due to emotional problems	Below average - due to emotional problems	Slightly more than average	Average	Poor	40%	Not applicable
OS	5th	Slightly below average - due to emotional problems	Average - no reason given	Slightly more than average	Slightly more than average	Slightly less than average	50%	Improved confidence
BA	6th	Below average - due to poor basic facts and slow computation methods	Slightly less than average - problem unfamiliar material	Average	Average	Average	50%	Improved motivation
BM	8th	Below average - due to a poor attitude	Below average - no reason given	Slightly more than average	Slightly more than average	Average	50%	Improved mathematical ability, source unknown
MB	8th	Below average - slow reasoning ability and poor habits	Below average - no reason given	Poor	Poor	Average	50%	No change noticed
SC	8th	Below average - due to excessive absenteeism	Slightly less than average - no reason given	Slightly more than average	Slightly more than average	Average	50%	Improved motivation

proposed causes of retardation. Computational difficulty, poor basic number concepts and motivational difficulties were all indicated.

Present Level of Non-Mathematical Achievement

All except two of the students were rated higher in non-mathematical achievement than on question one; but none of those were above average. The other two students were rated at the same level in this area as they were in mathematics. These results are not surprising, since visually handicapped students are usually lower in mathematical achievement than in other subjects. Only two of the teachers gave any reasons for below average performance in these other areas. But neither was very helpful; the fourth grade boy had an attitude problem and the sixth grade girl had difficulty with unfamiliar material.

Mathematics Learning Ability

Only three of the students were rated lower than average: the boy from the special education class, the fourth grade boy and one of the eighth grade girls. Their progress in computer programming seems to belie two of these ratings. Considering his emotional difficulty, the progress of the boy from the special education class was acceptable, if below average. Both the fourth grade boy and the eighth grade girl performed very well, although identified as slow learners. In fact, the eighth grade girl finished the course with ease.

General Adjustment and Prognosis for Job Success

The answers to the adjustment question were that the older children, sixth grade and above, were of average adjustment while the younger children were below average. This agrees with the relative emotional development of the children, as pointed out in the replies to the other questions. None of the students were given more than an even chance at successfully holding employment. The younger students were generally given lower ratings. This is also consistent with the other findings.

Lesson Observations and Statistics

Lesson One: The Four Fundamental Operations

The students had not been told anything about what they were going to do until they arrived for their first lesson. This was done to avoid any unnecessary apprehension on their part. Since the students did not understand the nature of the project, they were initially afraid to make errors, as reflected by their dependence on the instructor for constant reassurance.

All except one of the students covered this lesson with ease, to include working eight simple calculation problems at the end of the lesson. The slower student was from the special education class and, as it turned out, he had just started learning multiplication. Even so, he was interested in learning how to use the computer and applied himself during the class period.

Lesson Two: Priorities of
Mathematical Operations

All of the children completed this lesson with relative ease, with the exception of the boy from the special education class. During this lesson the eighth grade girls began to progress more rapidly than the others. They covered the entire lesson, including the parentheses exercise, and had sufficient time at the end to formulate their own applications of what they had learned. These girls had another advantage in that they all knew how to type. They were not hampered by having to search the keyboard with their fingers, or in some cases, with their limited vision. As the lessons progressed, this advantage was somewhat negated by the fact that the other children became more adept at remembering key locations.

The student from the special education class spent more than half of the second lesson reviewing the first lesson. By the end of lesson two he had had instruction on all of the material, but had just begun its applications. He was more relaxed during this lesson, however, and did not lose any of his motivation.

This was the last lesson for one of the fifth grade boys. He decided to drop out of the classes and no effort was made to stop him. He was told that he could rejoin the classes within a week; he did not. His teacher indicated that he was simply in a stubborn mood. The subject had done well in both lessons but subsequently told the instructor that he was "bored" with computer programming. The two fifth grade boys were good friends and the one that dropped the course tried to

talk the other boy into dropping as well. In spite of this influence, he completed the lessons.

Lesson Three: Line Numbers Combined with the GO and ERASE Commands

At this point, the actual lessons began to lag the planned lessons. There were two exercises (see Appendix K) planned for the end of lesson three. Two of the eighth grade girls were able to complete the entire lesson. All of the other students finished one of the exercises, except for the boy from the special education class. He spent the majority of this lesson in review of lessons one and two and was able to receive only a brief resume of lesson three. His slower progress seemed to be a result of his emotional makeup. He was very interested in the computer but needed constant reassurance that he was going to perform each step correctly. The need for reassurance had continually decreased in the other children, but not in this student. By the end of most of the lessons he appeared to be very restless; nevertheless, he always stated he wanted to continue. His actual level of motivation was extremely hard to judge because he was very reserved. Due to a lack of progress on his part and a concern for his well-being on the part of the instructor, the student was removed from the course after lesson five.

Lesson Four: SET, DO, FOR and QUIT Commands

Lesson four started badly for most of the students because they had one more example to work from lesson three. The example seemed

important enough to not omit. By the time they had completed it there was insufficient time remaining to cover lesson four. Two of the eighth graders spent their entire fourth period without any time to work on their own. This was not representative of their abilities, but can be attributed to the particular situation on the day the lesson took place. The fact that they improved on subsequent lessons corroborates this. All except the special education student worked the exercises that were included as part of the lesson symopsis. Again, these exercises helped the instructor evaluate the students' relative progress through the lessons.

Lesson Five: ASK and IF Commands

Performance on this lesson was much the same as on lesson four. The students were behind when they started and consequently were unable to complete it as planned. It was evident at the end of the session that some of the students would be able to finish in one more session. The sixth lesson was added to give that opportunity to as many as possible. As a result, the three eighth grade girls finished the material covered by the planned lesson five while the three remaining students (fourth, fifth, and sixth graders) all completed a portion of it. None of these remaining students went beyond the number multiplication exercise mentioned in the lesson description and they did not complete it until lesson six.

Lesson Statistics

The length of each lesson was recorded at its conclusion. The length of time the instructor spent teaching or reviewing was also recorded. It was not originally anticipated that there would have to be any significant amounts of review, since the lesson synopses were furnished for home study purposes. Consequently, the time spent in review was not separately measured. Scheduling and transportation difficulties caused the class periods to be of unequal lengths, making it difficult to compare the students' achievements. The ratio of instructor time to total class time is listed in Table 3. It is a measure of the relative amount of instruction each child needed for roughly equivalent amounts of material, and is an indication of the progress of each child through the six lessons. The total amount of time for each lesson is listed in Table 4. Examination of these two tables leads to the following generalizations about the project:

1) Table 3 shows that succeeding lessons become increasingly difficult, as indicated by the steadily increasing ratio of instruction time to class time. Although all of the students started with approximately the same percentage of instruction time on the first lesson, the younger students (sixth grade and below) required more instruction on the final lessons and consequently had less time to spend on individual work. This suggests that the younger students had a more difficult time learning programming but not a more difficult time learning calculation. This agrees with the instructor's observation, too.

Table 3. Ratio of Instructor Time to Total Class Time

Initials	Grade	Lesson Number					
		1	2	3	4	5	6
PM	Sp. Ed.	.73	.82	.90	.87	1.0	
IG	4th	.50	.68	.72	.78	1.0	1.0
ES	5th	.66	.53				
OS	5th	.52	.68	.73	.86	.98	1.0
BA	6th	.58	.69	.66	.78	.87	1.0
BM	8th	.48	.51	.43	1.0	.88	.70
MB	8th	.49	.46	.60	.72	.75	.88
SC	8th	.62	.70	.75	1.0	.89	.89

Table 4. Total Class Time per Lesson

Initials	Grade	Lesson Number						Total	Average class time/student
		1	2	3	4	5	6		
PM	Sp. Ed.	42	60	52	46	45		245	49
IG	4th	51	50	70	65	45	55	336	56
ES	5th	57	39					96	48
OS	5th	46	33	55	56	47	43	280	47
BA	6th	43	38	53	39	42	40	255	43
BM	8th	45	53	70	15	52	36	271	45
MB	8th	34	48	55	52	48	46	283	47
SC	8th	47	32	48	39	40	46	252	42
TOTAL		365	353	403	312	319	266		377
AVERAGE CLASS TIME/LESSON		46	44	58	45	46	44	283	47

2) Table 3 and Figure 3 illustrate the relative progress of the students through the lessons. Since each lesson had a fixed amount of material to cover, the percentages of time spent teaching in any given period are a measure of progress. In the final classes the planned amount of material was not finished by the younger students as indicated by the one hundred percent teaching times; therefore, those students were not able to learn FOCAL programming in six lessons.

3) Table 4 shows the average length of a lesson was 47 minutes and over the six periods each student's average was very close to the mean. The fourth grade student varied the most from the mean, but only because he was available for a longer period of time than the other students.

Time Spent in Review

Figures 4 and 5 are graphs that show the relative progress and amount of review each student needed for each class period. It represents the (subjectively estimated) relative amount of material covered on the ordinate and the class periods on the abcissa. The dashed line represents the originally expected rate of progress through the lessons with no review. It was expected that review would be done as homework by the student. A decreasing vertical line, i.e., no progress, beyond the previous level, represents review, and its length gives an indication of the amount of material that was reviewed. It is also indicative of the amount of time spent in review since generally it takes more time to review a large amount of material than a small amount.

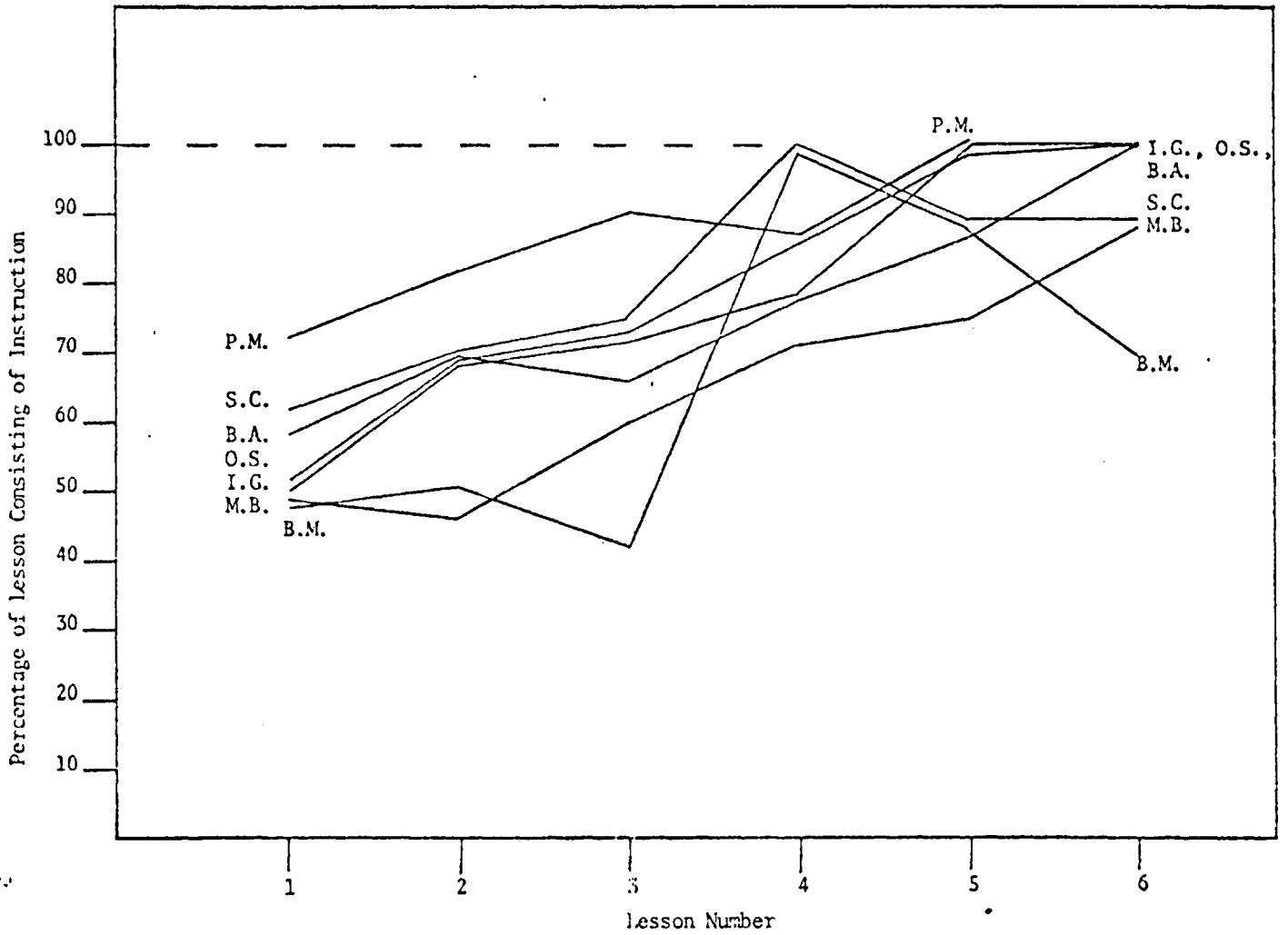


Figure 3. Instruction Time Ratio for Each Lesson.

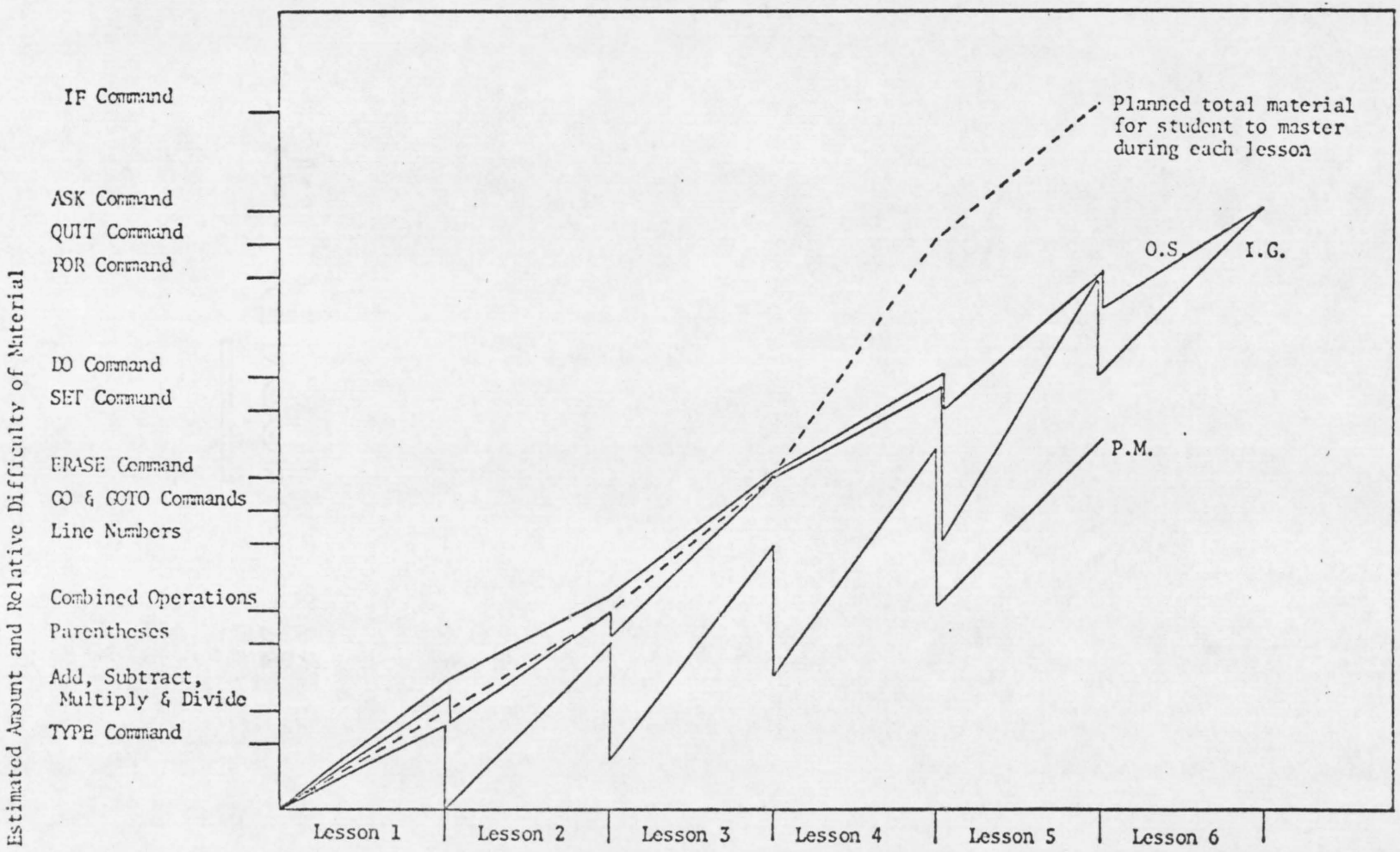


Figure 4. Review Graph, Elementary Students.

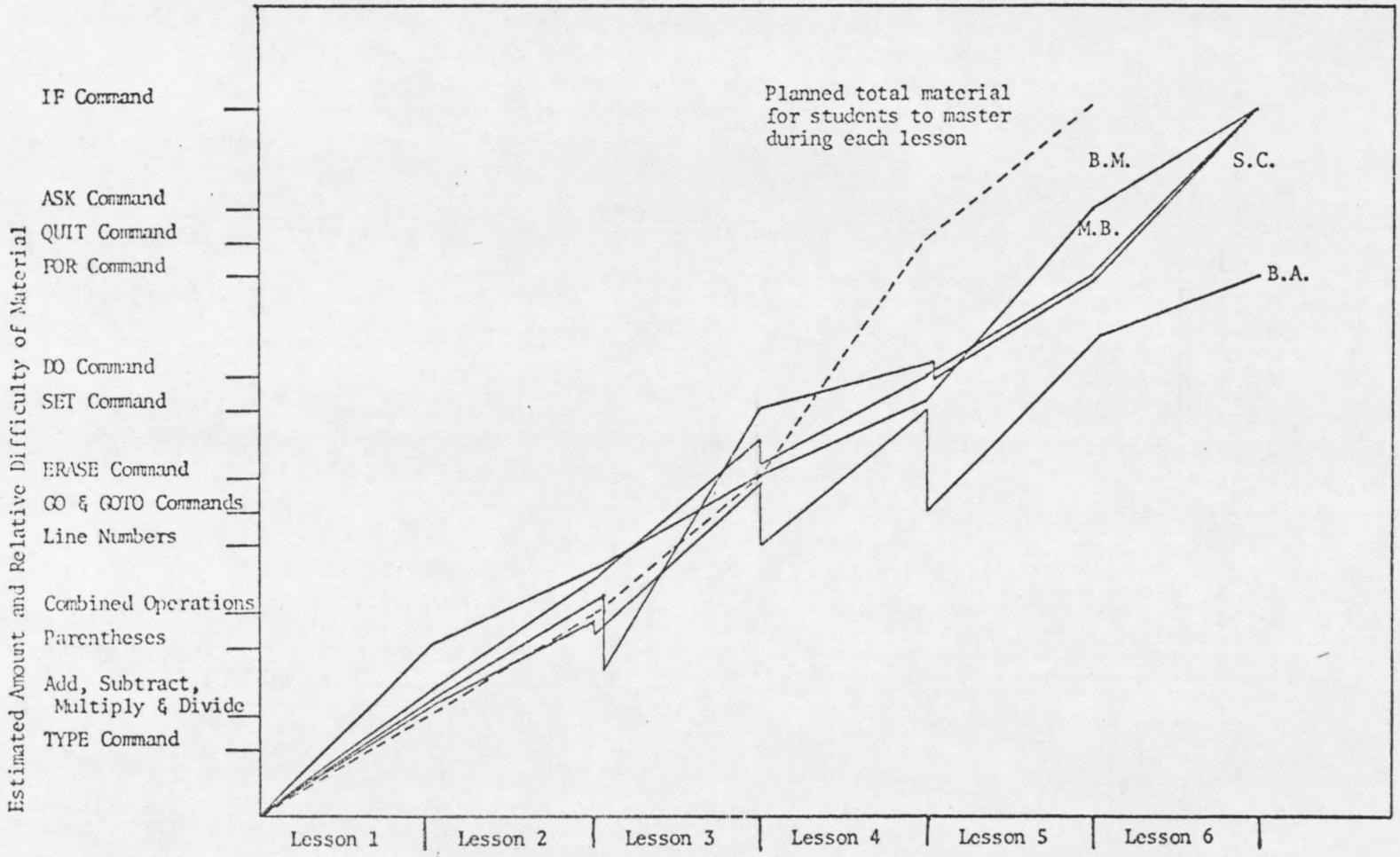


Figure 5. Review Graph, Junior High Students.

The consequence of review is to drop the student below the line of expected progress.

Figures 3 and 4 also illustrate the greater success of the eighth graders; they not only completed the lessons but required less review than the others. The figures also indicate that additional lessons should be added for the younger students. These additional lessons should probably be added between lesson three and lesson six and could be so structured as to contain more planned review and application of previous material

Teachers' Post-Instruction Evaluation

A sample evaluation form is in Appendix I. It simply asked for comments on any noticeable effects of computer programming.

Special Education Boy

The special education teacher stated that her student benefited from the personal contact he had with the programming instructor. She thought it improved his self-esteem in being able to relate to someone outside of his normal environment. She did not notice any particular changes in his academic endeavors; however, considering his progress in the lessons, that was not unexpected. In fact, it is felt that this student benefited emotionally from his experience more than he did intellectually. As stated earlier, he was much less reserved with the instructor at the end of his lessons.

Fourth Grade Boy

The fourth grade teacher did not notice any changes whatsoever in her student.

Fifth Grade Boys

The fifth grade teacher was very helpful by enlightening the instructor about the past behavior of the student who dropped the course, even though she was unable to determine any concrete reasons for his dropping. She did say that she felt he regretted his decision afterward but was too stubborn to change his mind. She said the student who completed the course was very enthusiastic about computers. He made statements to her that using the computer was much easier than using a type slate. She said he has difficulty with computation in class and when he makes an error he now tells her he would be able to work the problem if he had a computer. This suggests that he has confidence in his ability to operate the computer.

Sixth Grade Girl

The sixth grade teacher noted that her student was enthusiastic about computer programming, but she did not know whether to relate it to programming or to the fact that the student was getting out of class to go to the programming lessons. The student told her that the lessons had helped her with arithmetic. If the student thought that, then her confidence had been boosted and she benefited from the experience.

Eighth Grade Girls

The eighth grade mathematics teacher noticed an improvement in ability in one of her students, in enthusiasm in another student and no change in the last student. The change of ability in one student could have been caused by the lessons or by the simultaneous improvement in the performance of her student teacher. This teacher also identified one of the more elusive aspects of this problem when she stated that she was interested in finding out if this change would be transient or permanent. If the lessons were a part of the school curriculum then the effects should conceivably be more nearly permanent due to the resulting increase in programming instruction. The second of this teacher's students was so enthusiastic about the lessons that she continually queried her teachers as to the date and time of her next lesson. This same student was so reserved with the instructor that he could not visibly tell whether or not she enjoyed her lessons. He was able to tell that she was learning as evidenced by her improved programming ability. The third student learned programming well also but apparently did not conspicuously transfer it to her daily work.

Evaluation and Summary

The information contained in the Teachers' Post-Instruction Evaluation is summarized in Table 2.

Post-Instruction Evaluation by the Students

The students were interviewed to determine their feelings about the lessons they had been given in programming. See Appendix J for a sample interview form and Table 5 for a summary of the results.

Table 5. Students' Post-Instruction Evaluation

Initials	Grade	Likes	Dislikes	Confidence in Prog. Ability	Social Considera-tions	Teacher's Influence	Any help in other classes	Fun
PM	SE	Learning about numbers	None	Remembered what he had done	Wanted to know about the computer	How was he doing	Thinks it might help (not sure where)	Yes
IG	4	Commands, mechanical aspects	None	Need a little help	Did not talk to anyone	No comments to him	Multiplication tables	Was okay
OS	5	Liked its calcula-tion speed	None	Try it by himself first	Did not talk to anyone	No comments to him	No	Yes
BA	6	No com-ment	None	Need a little help	Others would like to do it too	Asked what she had learned	Typing	Yes
BM	8	Easy to Learn	Lines too far apart	Need a little help	Friends were impressed a lot	No comment to her	Arithmetic	Yes
MB	8	Fun to learn	None	Need a little help to get started, then go by herself	Did not talk to anyone	Wanted to know if hard and how she liked it	Typing	Yes
SC	8	No com-ment	None	No help needed	Did not talk to anyone	Thought programming was fun	No	Yes

Likes and Dislikes

The first two questions dealt with their likes and dislikes about computer programming. Their likes outweighed their dislikes significantly and included: "learning about numbers"; "liked the commands"; "liked the speed of doing math problems"; "fun to work"; and "easy to learn." Only one dislike was registered and that was that the spacing between the lines printed out in braille was too great. A simple software alteration could change that, if necessary, but the spacing is there to enable the student to read a line immediately after it is typed.

Confidence

A measure of the students' confidence in ability to use what they learned was obtained by asking whether or not they could write a program similar to those done in class, with no help, a little help or a lot of help. Three of the students stated they could try it on their own, and two of those thought that they might need a little help along the way. The rest of the students said that they would need a little help and none of them indicated a lot of help would be needed. Regardless of what they would actually need, they were all confident in their ability, a major step toward motivation.

Social Considerations

To determine whether or not the students were enthusiastic about the project or staying in it just because of an increased social position among their peers, or a similar motive, the students were asked what their friends thought about the lessons they were taking. Only three of the seven had even talked to their friends about the lessons and their

replies were: "friends are impressed a lot"; "friends would like to learn programming"; and "friends wanted to know about the computer." This may have had some effect on decisions to do the lessons but was not thought to be a major source of motivation for the group as a whole.

Teacher Influence

Next, the students were asked what their teachers had said about their learning to program, to determine if there had been any influence from the teachers. The answers indicated nothing more than interest on the part of the teacher and certainly no attempt to persuade the student one way or the other, with the exception of the fifth grade boy who was counseled by his teacher when he said he was going to drop out after his friend had. The same boy did not report his teacher's influence when questioned at the end of the program. Perhaps he wanted the decision to have been his own since he appeared to enjoy the lessons.

Carryover and Enjoyment

The final question asked if the student had learned anything that would benefit him in other classes; and if the lessons were fun. Only two of the seven students polled said that the lessons were of no help in any other class. The lessons were reportedly most helpful in typing and arithmetic classes. In reality, the instruction did not last long enough to be of any significant help, but the students did recognize a possible carryover. There might be a carryover of enthusiasm as well, especially since all of the students thought the lessons were fun.

CHAPTER 7

CONCLUSIONS

The project hypotheses have been defined on several levels. On the upper level are the four primary hypotheses regarding the causes of poor mathematical achievement in visually handicapped children. Testing these four hypotheses was not within the scope of this study. The second level of hypotheses dealt with computer applications that might help alleviate the problem of low mathematical achievement. The study was also too limited to test these hypotheses; however, it did undertake an investigation of the problems connected with teaching young blind students to program a computer. This was the third level of hypotheses. It was necessary to answer these questions before any further study could be done on the hypotheses regarding the application of computers to the problem of low achievement.

Calculating for Young Students

Even very young students, fourth graders in this project, can be taught to use a computer as a versatile calculator. All but one of the students were proficient in this application after two lessons. The boy from the Special Education class was somewhat slower; however, he was able to learn calculating. This implies that emotional problems alone may not exclude a child from such a period of instruction, even though they have led to severe underachievement in the past.

Another indication is that it may be possible to use a computer calculator at the early ages when the number concepts are still being formed. It was also shown that typing experience is not a prerequisite, especially since all of the students eventually learned the layout of the keyboard.

Programming for Young Students

No student below the eighth grade completed the course. In other words, the children from the lower grades had a harder time learning the elements of programming. They did not have trouble with the mechanics of programming, rather it was with the algorithmic thinking ability that programming requires. This may be an indication that their instruction should have been organized differently, e.g., more time spent on each programming technique, etc., or it may be that at this stage in their development, they are not prepared for algorithmic thinking techniques. This raises an important issue for the problem of designing instruction in computer use.

Motivation

All of the students exhibited pride and enjoyment in their accomplishments to such a degree as to indicate that the hypothesized advantage of computer usage in building motivation is quite probable. They were all low achievers in mathematics; conversely the programming language and all of the applications were mathematically oriented. Nevertheless, seven of the original eight students continued to the end of the project and all of them wanted to return for one additional programming experience at the end.

The greater motivation was not due to an increase in social status with peers, but to the intrinsic aspects of the computer teaching situation. One of those aspects was most probably due to the close relationship between student and instructor; however, this relationship was not the only source of their high morale as shown by the fact that the students opted to do programs on their own in the final session.

Individual Instruction

Close individual supervision is probably necessary at first for teaching this kind of programming. The student has inhibitions about interacting with the computer and he has a fear of making mistakes. To ensure the motivation necessary for a good start, individual attention is required. After completion of the beginning lessons, much of the remaining teaching could be done in the classroom environment, provided contact with the machine is maintained at frequent intervals. This contact increases the motivation as well as provides an opportunity for the student to apply the principles he has learned in the classroom.

Students' Fear of Errors

The students' fear of errors and consequent hesitancy to work on their own is an obstacle to their using the computer to explore mathematics. This obstacle is not caused by their past interactions with the computer, since those interactions were always positive. The instructor observed that the hesitancy became less pronounced as the students became more experienced and began to master the operation of the computer.

Homework

Some of the learning can be done as homework, and review is necessary to rapid progress, but homework should contain only essential information to avoid becoming boring, thereby risking loss of motivation and interest.

Measurement of Student Achievement

Future attempts to teach computer programming to visually handicapped children should be founded on a more reliable and objective means of measuring student progress. Such measures must allow an accurate comparison of all grade levels while being free from bias and anxiety producing material.

Non-Mathematical Application

More non-mathematical applications, e.g., games, might help keep the interest of the children over a longer period of lessons. Strikingly, though, the emphasis on mathematics in this project did not appear to affect motivation, or was perhaps offset by the novelty and fun of using the machine. This is important for a project directed toward teaching mathematics.

Hardware Related Conclusions

The teletype adaptations previously described provided adequate output for calculations as well as for later review. The durability of the output was not as high as that from standard braille equipment, e.g, braille writer.

The PDP-8/e had sufficient core memory, i.e., 4K, to permit interactive programming. It would be possible to teach similar lessons using one or more time-shared computer terminals. The projected usefulness of a computer teaching terminal at ASDB has not been determined; it of course depends on whether the hypotheses about the computer's effect on the mathematical achievement of the blind are correct.

Interest Carry Over

There are slight indications from teachers' remarks that there may be a carry over of interest from computer programming instruction to mathematics applications; however, the indications are not at all conclusive.

APPENDIX A

HARDWARE MODIFICATIONS

With the below listed exceptions, more information concerning the modification of the Teletype Corporation Model 33 Teletype may be obtained from the Anderson and Rogers article, "An Inexpensive Braille Terminal Device," (1968).

Print Head Grinding

As soon as the print head can be tested with the modified platen, a visual examination of the braille characters should be made to determine the uniformity of the character depth. Normally the single pins tend to punch holes in the paper while the triple pins emboss too shallow an indentation. This is corrected by grinding the single pins to a shorter height than the triple pins and then slightly rounding their shoulders. On the other hand, the triple pins should be more pointed and the double pins somewhere in between. Testing the head on the teletype occasionally as grinding progresses will aid in determining the proper height and sharpness of each pin.

Platen Modification

Anderson and Rogers (1968) recommended covering the platen with surgical rubber tubing one inch in diameter and 1/16 inch thick. Since that size tubing is not common and was not available, 1/2 diameter by 1/16 inch thick surgical tubing was used successfully, but it

was necessary to obtain a spare platen since the modification was permanent. The hard rubber backing is stripped off of the metal cylinder. This can be done by inserting a knife between the body of the platen and the rubber and then sliding it the length of the platen. The next step is to cut two 12 inch long pieces of surgical tubing lengthwise so that they can be placed over the surface of the platen. The cut should be as straight as possible to enable a close joint between the two pieces of tubing. The tubing is glued to the platen one piece at a time using rubber cement. After it has dried thoroughly, the excess tubing is trimmed off of each end of the platen. The platen is then ready for use.

Keyboard Modification

As previously mentioned, there is a clear plastic embossing tape available for the Dymo braille embosser. If the braille cells are punched out of the tape using a standard hole punch they are the right size for the keys. When brailling a key that has both upper and lower case it is necessary to place the lower case character on the left of the key and the upper case character on the right. This is caused by the greater height of the braille cell in comparison to its width. When keys have words on them, two letters can be used as a memory cue, e.g., "SH" for the "Shift" key and "RO" for the "Rubout" key. Unlike the print head and platen modifications, once this one has been completed it does not need to be removed for print operation.

Hammer Modification

This modification may not be necessary and should not be made until after the modified print head and platen have been tested together. If the teletype embosses the braille well the hammer modification is not needed. In this case, the braille was not embossed well so a small rubber pad was added to the pad already present on the hammer. It was made by cutting the end off of a rubber test tube stopper and then taping it onto the hammer with plastic electrical tape. This is another modification that does not need to be removed when normal print is being used, since it simply makes the print a little darker.

APPENDIX B

SOFTWARE MODIFICATIONS AND SHORTCOMINGS

Loading the Braille and Line Feed Subprograms

The PDP-8/e computer used for this project has a magnetic tape reader as an integral part of its system. This simplifies and speeds the process of loading the FOCAL program with the braille and line feed subroutines. Initially the computer has three programs in memory, the Read in Mode (RIM) Loader, the Binary (BIN) Loader and the magnetic tape READ routine. FOCAL is then read from magnetic tape into the computer memory. After this has been completed, the FOCAL initial dialogue begins. The first question of the initial dialogue is, "SHALL I RETAIN LOG, EXP, ATN?:" This question is answered with a "NO," thereby making available approximately 200_{10} more locations in core. The second question in the initial dialogue is, "SHALL I RETAIN SINE, COSINE?:" This question is answered "YES," thereby protecting the SINCE, COSINE portion of core from being erased during programming. This is the core location where the braille and line feed subroutines are overlayed. The two subroutines are written in Program Assembly Language (PAL III) and subsequently assembled into binary object paper tapes. These paper tapes are then read into core using the BIN Loader. Next the magnetic tape WRITE program is read into the core where BIN was previously located, and then FOCAL, with the braille and line feed subroutines, is recorded onto

the magnetic tape. The extended functions are no longer available, due to the limited size of the memory.

Braille Subprogram Shortcomings

There are two minor problems with the current braille subroutines but they do not cause any significant difficulties in operation.

Character Counter

The braille subroutine does not have a character counter to tell the computer when the student has gone beyond the end of a line. However, since there is a counter on the FOCAL input buffer, it is probable that the index value of this counter could be changed from 72 to 24. At present, the computer will accept everything beyond the end of the line until 72 characters are input. This is a definite advantage when long instructions are required. Of course, when the end of the line is reached everything subsequent to the 24th character is embossed in the same location and it cannot be read. However, the end of the line warning bell works and that helps the student keep track of his location.

Characters Not Printed

The current braille routine is bypassed by the first character of an error code. For example, error code "?01.40" translates as, "Illegal step or line number used." The question mark is missed by the braille routine and the computer attempts to type it in normal print. However, since there is nothing on the braille print head in the question mark location, the result is "01.40." Since all of the error codes

start with a question mark and have the same number of digits, there is little or no difficulty recognizing errors when they occur.

A similar event occurs on those error codes that have line numbers following them. For example, if the normal error code were "?01.40 @ 07.12" this would indicate that the error occurred at line number 07.12. However, the braille routine does the same thing with the "@" sign that it does with the question mark and the result is nearly the same. In this case, there are pins on the print head near the "@" sign location, so a miscellaneous one-half braille cell is embossed before the address. Here again, the only time this occurs is on an error code that includes a line number, so it does not cause any difficulty in recognition.

APPENDIX C

BRAILLE SUBPROGRAM

The following is the Program Assembly Language (PAL III) listing of the Braille Subprogram (Programming Languages, 1972).

```
*167
SUB,BEGIN
*5200
707
114
307
1016
105
112
110
716
1410
1014
1612
116
103
1616
107
1216
1416
703
714
303
1403
1603
314
1414
1614
316
1412
1407
1605
1010
516
1012
1207
712
705
```

```
1212
512
312
1205
505
305
1203
503
701
710
1201
501
301
1210
510
310
1214
514
1601
1610
1003
101
1001
1401
1607
1405
1005
507
1007
BEGIN,0
DCA TEMP2
TAD TEMP2
SPF
TAD K240
SMA
JMP BRAIL
TAD J240
TSF
JMP .-1
TLS
JMP I BEGIN
K240,-240
J240,240
BRAIL,TAD K5200
DCA TEMP
TAD I TEMP
AND MASK
TAD K300
TSF
JMP .-1
TLS
```

```
CLA
TAD I TEMP
BSW
AND MASK
TAD K300
TSF
JMP .-1
TLS
CLA
TAD K307
TSF
JMP .-1
TLS
CLA
TAD TEMP2
JMP I BEGIN
TEMP,0
MASK,77
K300,300
K5200,5200
K307,307
TEMP2,0
*1355
JMS I SUB
*2711
JMS I SUB
*2762
JMS I SUB
*4465
JMS I SUB
*4514
JMS I SUB
*2615
JMS I SUB

$
```

APPENDIX D

LINE FEED SUBPROGRAM

The following is the Program Assembly Language (PAL III) listing of the Line Feed Subprogram (Programming Languages, 1972).

```
*0063
OUTDEV,2676
*0076
CLF,0212
*0002
SLF,FEED
*5354
FEED,0
TAD PARAM
DCA PARAM1
REDO,TAD CLF
JMS I OUTDEV
ISZ PARAM1
JMP REDO
TAD CLF
JMP I FEED
PARAM,7770
PARAM1,0
*2500
JMS I SLF
$
```

APPENDIX E

LESSON SYNOPSES

Lesson One Synopsis

1. Arithmetic signs

   Addition (+)

   Subtraction (-)

   Multiplication (*)

   Division (/)

2. Arithmetic examples

   1+5=6

   7-3=4

   4*2=8

   9/3=3

   Notice that the computer does not use a number sign.

3. Commands

   TYPE - tells the computer to work your problem and then type out the answer.

4. Examples

   *TYPE 1+5 (push return key for answer)

   =6*

   *TYPE 7-3 (The asterisk at the beginning and the end are not multiplication signs. They tell you the machine is ready for you to do your next problem.)

= 4*

*TYPE 4*2 (The second asterisk is a multiplication sign.)

= 8*

*T 9/3 (Each of the commands can be abbreviated. In this case T stands for TYPE.)

=3*

Lesson Two Synopsis

1. Use of Parentheses

Parentheses tell the computer what to do first. If you have a long problem and do not use parentheses, the computer will multiply first, divide second, add third and subtract last.

Example 1

Add 4 plus 4 and multiply the answer by 2.

Without parentheses:

*TYPE 4+4*2

= 12*

With parentheses:

*TYPE (4+4)*2

= 16*

The problem with the parentheses is the correct answer. The computer always does the operation inside the parentheses first and each open parenthesis must have a closed parenthesis somewhere in the problem.

Example 2

Add 3 to 7 and multiply the answer by the sum of 4 and 9.

Without parentheses:

*TYPE 3+7*4+9

= 40*

With parentheses:

TYPE (3+7)(4+9)

= 130*

Lesson Three Synopsis

1. Examples of Line Numbers

1.01 Lowest possible line number

1.02

1.03

1.04

1.10

1.20

1.30

1.40

4.51

4.52

4.53

4.54

31.99 Largest possible line number. Note: The number before the period is the subroutine number and the number after the period is the instruction number.

2. GO command: When the command GO is given, the computer starts working at the smallest line number and continues to the end of the program.

3. GOTO command: When the command GOTO is given, the computer starts working at the line number after the GOTO command and continues to the end of the program.

   Example:

   GOTO 7.2 (The computer will start working on line number 7.2.)

4. ERASE command: Tells the computer to forget the line numbers after the ERASE command.

   Examples:

   ERASE 1.01 (The computer will forget line number 1.01)

   ERASE 2.0 (The computer will forget all line numbers starting with a 2. In other words, the entire second subroutine.)

   ERASE ALL (The computer will erase all subroutines in the machine.)

Lesson Four Synopsis

1. Control Characters:

   a. ! - returns the printer to the left margin and rolls the paper to the next line.

      Example:

```
*TYPE 11-3,!,2+4
= 8
= 6*
```

   b. " - placed at the beginning and end of words that you want the computer to type.

Example:

```
*TYPE "THE ANSWER IS",1+4
THE ANSWER IS=5*
```

c. % - (Eighth graders only) - changes back and forth between decimal numbers and integers.

Examples:

```
*TYPE %6.2,8/3
= ---2.67
```

In this case, the answer uses 6 spaces with 2 of the spaces to the right of the decimal.

```
*TYPE %6.0,8/3
= -----3
```

In this case, the answer has six spaces but there are no spaces to the right of the decimal. (Note: the answer is rounded off to the nearest integer.)

2. Commands:

a. SET - Tells the computer to remember a name for a number

Examples:

```
*SET A=4
*SET B=10
*SET C=A*B
*TYPE C
= 40*

*2.1 SET BOY=105
*2.2 SET GIRL=15
*2.3 SET SUM=BOY + GIRL
*2.4 TYPE "SUM", SUM!

*GO
SUM = 120*
```

```
*3.1 SET DIVIDE= BOY/GIRL
*3.2 TYPE "DIVIDE", DIVIDE
*GO
SUM= 120
DIVIDE= 7*
```

b. DO - Used to tell the computer to do only a part of the program. If you say DO 2.0 in the above program then only the second subroutine will be done. If you said GOTO 2.1, then the second and third subroutines would be done.

QUIT - tells the computer to stop working.

Example:

```
*4.1 DO 7.0
*4.2 DO 8.0

*7.1 SET A=10*1
*7.2 SET B = 10*2
*7.3 SET C=10*3
*7.4 SET D=10*4
*7.5 SET E=10*5
*7.6 TYPE A!,B!,C!

*8.1 TYPE D!,E
*8.2 QUIT (This command tells the computer to stop working.)

*GO
= 10
= 20
= 30
= 40
= 50*
```

*4.11 QUIT (If this statement was added then 4.2 would never be done.) For example:

```
*GO
= 10
= 20
= 30*
```

c. FOR - Used to tell the computer to do something more than once and to change some value each time it is done.

Example:

```
*1.01 FOR I=1,1,5;DO 2.0
*1.02 QUIT

*2.1 SET A=10*I
*2.2 TYPE A,!

*GO
= 10
= 20
= 30
= 40
= 50*
```

Notice that this program does the same thing as the last program, but it is much shorter.

The first number after the equals sign is the starting value of I, the next number is added to the first number each time the computer completes the instruction. The computer keeps going until the value of I equals the last number.

Lesson Five Synopsis

1. Commands

a. ASK - Used when the computer needs information from the programmer. The computer stops when it comes to an ASK command and waits for information to be typed on the keyboard.

Example:

```
*1.1 TYPE "WHAT NUMBERS DO YOU WANT TO MULTIPLY?"!
*1.2 ASK A
*1.3 ASK B
*1.4 TYPE A*B,!
*1.5 GOTO 1.1
```

This program is called a loop because it will keep repeating until it is stopped from the control panel. If line number 1.5 was a QUIT command, then you would have to restart the program after each run.

b. IF - This command is used to tell the computer where to go depending on whether the number in the parentheses is negative, equal to zero, or positive.

Example: Triangle Selector Program

```
*1.1 TYPE "WHAT IS THE LENGTH OF THE BASE?"!
*1.2 ASK B
*1.3 IF (B-20)1.4,1.5,1.6
*1.4 TYPE "SMALL TRIANGLE";GOTO 1.1
*1.5 TYPE "MEDIUM TRIANGLE";GOTO 1.1
*1.6 TYPE "LARGE TRIANGLE";GOTO 1.1
```

This program will tell you which of the three plastic triangles you have when you type in the length of the base of that triangle. When you type in the length of the base, the IF command subtracts 20 from the length. If the length was 10 to start with then 10-20 is negative and the program goes to line number 1.4. If the base was 20 to start with, then 20-20 is zero and the program goes to line number 1.5. If the base was 30 to start with, then 30-20 is positive and the program goes to line number 1.6. This program is also a loop that continues until it is stopped from the control panel.

c. WRITE - This command is used to make the computer type out any or all of the instructions that have been placed into the computer using line numbers.

Example: (From the above program)

```
*WRITE 1.1
01.10 TYPE "WHAT IS THE LENGTH OF THE BASE?"!*
```

If the command had been *WRITE 1.0, then the entire first subroutine would have been typed out. If the command had been WRITE and no line number after it, then the entire program would be typed out. In other words, all of the subroutine would be typed. This command is used to check your programs and it helps when you are looking for errors.

APPENDIX F

INTRODUCTORY MESSAGES

Lesson One: The Four Fundamental Operations

Hello, I am a computer. I want to help you with your arithmetic. I am very easy to use and lots of fun too. Jim is going to show you how to tell me what to do. So you listen carefully and I will be a very good friend of yours.

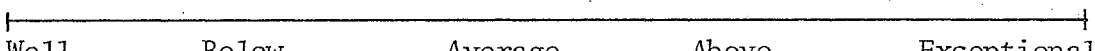
Lesson Two: Priority of Mathematical Operations

Hello again from your computer friend. You did so well in your first lesson that Jim has decided to teach you something really exciting today; they are called parentheses. There is an open parenthesis above the number eight and a closed parenthesis above the number nine. Now it is Jim's turn to show you how to use them.

APPENDIX G

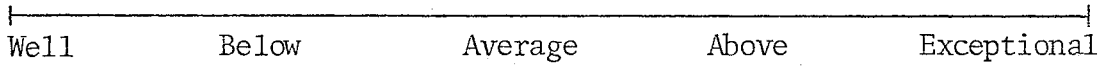
TEACHERS' PRE-INSTRUCTION EVALUATION

1. Please draw an arrow on the graph to indicate the present mathematics achievement of this student in relation to all other blind students, at this grade level in your experience.



2. If the student was rated below or well-below average, do you attribute it to any specific difficulty and if so, what? (e.g., computation, logic, attention span, etc.)

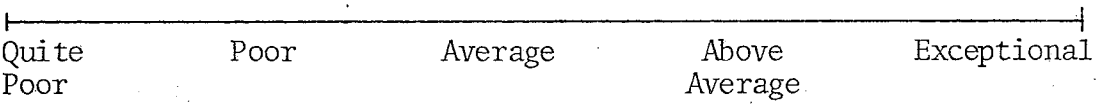
3. Please draw an arrow on this graph to indicate the student's present level of achievement in those areas of study other than mathematics. Once again, this should be in relation to all other blind students at this grade level in your experience.



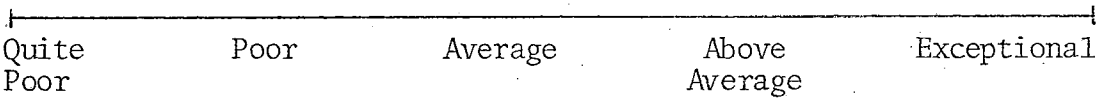
4. If the student was rated below or well-below average do you attribute it to any specific difficulty other than those in question 2, and if so, what are they?

5. On the next graph please indicate the learning ability of student in each area of study listed.

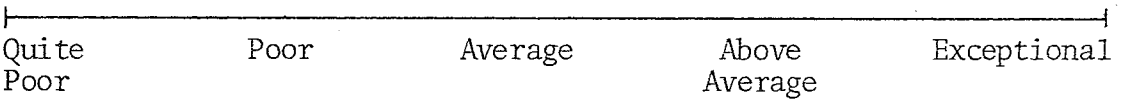
Mathematics:



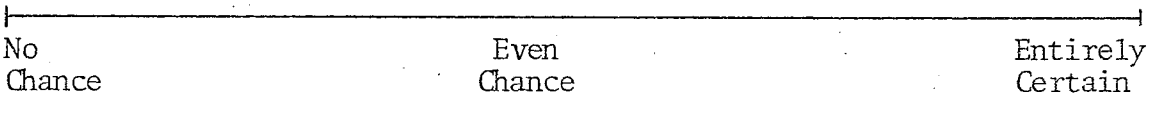
Other subjects:



6. On this graph please indicate the student's general adjustment with respect to all other blind students at this grade level in your experience.



7. What would you guess is the likelihood that this person will eventually successfully hold a job? Please indicate your response on the graph.



8. Does this student have any special problems that I should know about, other than those already listed?

9. Is this student congenitally blind and if not, how long has he or she been blind?

APPENDIX H

PROGRAMMING INSTRUCTOR'S EVALUATION

1. Was extensive review necessary?
2. Did the student seem to understand the concepts presented?
3. How much of the lesson plan was covered?
   a. Starting point from previous class:
   b. Ending point:
4. Questions asked by student:
5. Was the student at ease and/or enthusiastic during the lesson?
6. Remarks (problems, interruptions, distractions, etc.):
7. Statistics:
   a. Total time elapsed:
   b. Total instruction time:

APPENDIX I

TEACHER'S POST-INSTRUCTION EVALUATION

I need your evaluation of the impact of computer programming on your student. I am interested in any contribution you are willing to make, including such areas as changes in abilities or attitudes, amount of interest or enthusiasm displayed, effects on peers, etc. Please give your reply careful consideration and accept my sincere thanks for your cooperation through the past weeks.

APPENDIX J

STUDENT EVALUATION OF INSTRUCTION

Now that you have finished your classes in computer programming, I want to know what you thought about them. Your answers will help me improve the classes in the future.

1. What did you like best about computer programming?
2. What did you like least about computer programming?
3. If I told you to write a program, do you think that you could do it by yourself, with a little bit of help or do you think that you would need a lot of help?
4. What did your friends say about your taking computer programming?
5. Did your teacher think that the programming lessons were a good idea?
6. Did you learn anything that might help you in any of your other classes?
7. Was computer programming fun?

If you are interested in doing more computer programming, I will help you to decide on a problem and write it during an afternoon here at school. When you have the program written then we will take it over to the University and try it out. If you decide that you would like to do this, then tell Miss B. before Friday so that I can make the arrangements.

APPENDIX K

LESSON EXERCISES

Lesson Two

1. Add 15 + 29 and divide the answer by the sum of 7 and 4.
2. Subtract 4 from 53 and multiply the answer by 37 minus 29.
3. Multiply 62 times 14 and subtract 56 divided by 8.
4. Divide 132 by 12 and add to it 14 times 77.
5. Add 31 to 159, then multiply that answer by 4 and divide the result by 20.

Lesson Three

1. Put into subroutine number one

   47 + 5

   35 - 17

2. Put into subroutine two

   72/9

   12 x 3

3. Tell the computer to work all of the problems.
4. Tell the computer to work the last problem only.
5. Tell the computer to forget the second problem and work the rest of them.
6. Tell the computer to forget everything and then test it to see if it did.

7. Put into subroutine number seven:

   Add 3 to 6 and divide by 9

   Subtract 7 from 27 and multiply by 2

8. Put into subroutine number twenty-one:

   Add 360 to 72 and divide by 4

9. Tell the computer to work all the problems.

10. Tell the computer to work only the last problem.

11. Tell the computer to forget the last problem and work only the second problem in subroutine seven.

12. Tell the computer to forget everything, and test to see if it did.

LIST OF REFERENCES

Anderson, Gary B. and David W. Rogers, "An Inexpensive Braille Terminal Device," Communications of the ACM, Vol. XI, No. 6, (June 1968).

Final Report to Social Rehabilitation Administration, Department of Health, Education and Welfare, The Sensory Aids Evaluation and Development Center, Massachusetts Institute of Technology, 31 Dec. 1970. Contract SRS-69-41.

Hayes, Samuel P., "Arithmetic: Teaching and Testing in Schools for the Blind," Outlook for the Blind, 1944.

Lewis, Marrian, "Teaching Arithmetic Computation Skills," Education of the Visually Handicapped, Vol. II (October, 1970).

Nolan, Carson Y., "Research in Education of the Blind," Blindness Research: The Expanding Frontiers, Maxwell H. Goldberg (ed), 1967.

Nolan, Carson Y. and Samuel C. Ashcroft, "The Stanford Achievement Arithmetic Computation Tests," Education of the Blind, Vol. VIII, (March, 1959).

"Programming Languages," PDP-8 Handbook Series, Digital Equipment Corporation, Maynard, Mass., 1972.

Riddle, Charles E., "What are the Most Retarding Factors in Teaching Seventh and Eighth Grade Arithmetic?" Proceedings, American Association of Instructors of the Blind, 1948.

Smith, Benjamin F., "Various Approaches to Mathematic Instruction: The Catherine Sterns Materials for Blind Children," Proceedings, American Association of Instructors of the Blind, 1964.