A Diagnosis of Beginning Programmers' Misconceptions of BASIC Programming Statements

PIRAYE BAYMAN and RICHARD E. MAYER *University of California–Santa Barbara*

This project was supported by the National Institute of Education, Program in Teaching and Learning, under Grant NIE-G80-0118. Richard Welker assisted in data tabulation. A more detailed report of this project is available from the authors [1].

Authors' Present Address: Piraye Bayman and Richard E. Mayer, Department of Psychology, University of California–Santa Barbara, Santa Barbara, California 93106.

0001-0782/83/0900-0677 75¢

ABSTRACT: In the process of learning a computer language, beginning programmers may develop mental models for the language. A mental model refers to the user's conception of the "invisible" information processing that occurs inside the computer between input and output. In this study, 30 undergraduate students learned BASIC *through a self-paced, mastery manual and simultaneously had hands-on access to an Apple II computer. After instruction, the students were tested on their mental models for the execution of each of nine* BASIC *statements. The results show that beginning programmers—although able to perform adequately on mastery tests in program generation—possessed a wide range of misconceptions concerning the statements they had learned. This paper catalogs beginning programmers' conceptions of "what goes on inside the computer" for each of nine* BASIC *statements.*

I. INTRODUCTION

The main focus of this paper concerns "what is learned" when a beginning user is taught a computer programming language such as BASIC. The outcome of learning can be viewed in two distinct ways: (1) Learning BASIC involves the acquisition of new information and new rules, such as when to use quotes in a PRINT statement or how to produce a conditional loop using an IF statement. (2) Learning BASIC involves the acquisition of a mental model, such as the idea of memory spaces for holding numbers.

This study explores the idea that the learning of BASIC involves more than the acquisition of specific facts, rules, and skills. Beginning programmers also develop mental models for the language in the process of learning the essentials of BASIC. Moran [6] suggests that the user develops a "conceptual model" of the system as he or she learns to use it. Moran defines the user's conceptual model as the knowledge that organizes how the system works. User models may not be accurate or useful representations of "what is going on inside the computer." However, most instructional effort is directed solely at helping the learner acquire the new information and behaviors without giving much guidance to the learner for the acquisition of useful mental models.

Mayer [3] has suggested a framework for describing the internal transformations that occur for elementary BASIC statements. In particular, any BASIC statement can be conceptualized as a list of transactions. A transaction is a simple proposition asserting some action performed on some object at some location in the computer. For example, LET D = 0 involves the following transactions: Find the number is memory space A. Erase the number in memory space A. Find the number indicated on the right of the equals sign. Write this number in memory space A. Find the next statement in the program.

TABLE I. Percentage of Users with Correct, Incomplete, Incorrect, and Empty Conceptions for the Nine BASIC Statements

Statements	Conceptions			
	Correct (%)	Incomplete (%)	Incorrect (%)	Empty (%)
INPUT A	3	30	30	37
30 READ A	10	27	17	47
IF A < B GOTO 99	27	27	40	7
LET A = B + 1	27	10	60	3
20 DATA 80, 90, 99	27	17	13	43
60 GOTO 30	27	56	10	7
PRINT C	33	0	47	20
LET D = 0	43	3	53	0
PRINT "C"	80	0	13	7

Experts and novices are likely to differ with respect to their mental models for programming statements. For example, an expert programmer may have developed an accurate conception for a counter set LET, such as the one just given. However, the novice may lack a coherent mental model or may possess incorrect conceptions for BASIC statements. In a recent study, Mayer and Bayman [5] found that novice and expert calculator users differed greatly in their conceptions of "what goes on inside the calculator" for various key presses.

How much training one needs to acquire a conceptual model (or mental model) has not yet been explored in research. A first step in addressing this issue is to study the novice user's understanding of newly learned programming statements. In this paper, we describe the novice user's conception of nine BASIC statements, using a transactional analysis.

II. METHOD

The subjects were 30 college undergraduates who had no prior experience with computers or computer programming. The subjects took a modified version of a self-instruction, self-paced, mastery course called *BASIC in Six Hours* [2] that is widely used for teaching BASIC in the Microcomputer Laboratory at the University of California–Santa Barbara. The instruction occurred over the course of three sessions and involved both a programmed manual and hands-on access to an Apple II computer. Subjects were required to pass mastery tests over one section before moving on to the next section of the manual.

Following successful completion of the course, the users were given a procedure specification test. For each of nine statements (from the instructional course), subjects were asked to write, in plain English, the steps that the computer would carry out for each statement. Users were instructed to write each step on a separate line of the test sheet. The method also involved additional tests which are described in more detail by Bayman and Mayer [1].

III. RESULTS

A. Scoring

Each user's protocol for each of the nine statements was broken down into a list of transactions by two scorers using a procedure described by Mayer [3].

Three types of transactions were observed for each statement: (1) *Correct transactions*—for example, the key correct transaction for LET A = B + 1 is "store the value of B plus 1

TABLE II. Percentage of Users Who Produced Key Correct and Incorrect or Incomplete Transactions for Nine BASIC Statements

Transaction	Percentage of [Users]
INPUT A	
*Print ? on the screen.	7
*Wait for a number and ⟨CR⟩ to be entered.	23
*Write the entered number in memory space A.	3
Write A in memory (or a data list).	30
Wait for a certain number or letter.	13
Print A on the screen.	3
30 READ A	
*Write next number from DATA in memory space A.	10
Print value of A on the screen.	10
Write letter A in memory.	3
Wait for number to be entered from keyboard.	3
IF A < B GOTO 99	
*If the value of A is less than B, then move to line 99 in the program.	63
*If the value of A is more than or equal to B, then move on to the next statement in the program.	33
Move to line 99 (without test).	10
Print the number 99 or line 99 on the screen.	10
Write A or B or A < B or a number in memory.	10
LET A = B + 1	
*Write the obtained value in memory space A.	30
Write the equation in memory.	43
Write B + 1 in memory space A.	33
Print A = B + 1 or A or the value of A on the screen.	23
20 DATA 80, 90, 99	
*Put numbers in input queue.	27
Put numbers in memory.	20
Print numbers on screen.	13
Put numbers in memory space A.	7
60 GOTO 30	
*Move to line 30 in the program.	67
*Continue from there.	37
Find the number 30.	10
Move to line 30 if A does not equal a certain value.	7
Print line 30 on screen.	3
PRINT C	
*Print number (or 0) on screen.	40
Print the letter C on the screen.	33
Write the letter C in memory	7
Print either "error" or nothing on the screen.	7
LET D = 0	
*Write zero in memory space D.	47
Write the equation in memory.	47
Write D or 0 in memory.	13
Print the equation on the screen.	7
PRINT "C"	
*Print the letter C on the screen.	83
Write the letter C in memory.	7
Print the value of C on the screen.	7
Find the number in memory space C.	3

Note: Asterisks indicate key correct transactions for each statement. The three most common incorrect or incomplete transactions are also listed for each statement.

in memory space A." (2) *Incomplete transactions*—for example, a user's answer for LET A = B + 1 could include "store the value of B + 1 in memory." (3) *Incorrect transaction*—for example, a user's answer for LET A = B + 1 could include "store the equation A = B + 1 in memory."

B. Differences Among Statements

In order to make comparisons of user conceptions among the nine statements, each user's verbal protocol for each statement was categorized as correct, incomplete, empty, or incorrect. The criteria for classifying protocols were as follows: *Correct*—if the user's protocol included the key correct transaction(s) and no incorrect transactions; *Incomplete*—if the user-produced incomplete versions of the key transaction(s) and no incorrect transaction; *Empty*—if the subject produced no correct or incomplete version of the key transaction(s) and no incorrect transaction; and *Incorrect*—if the subject produced one or more incorrect transactions. Table I presents a summary of the percentage of users who produced each type of conception for each of the nine BASIC statements. Table I presents the nine statements in order of difficulty based on the proportion of correct conceptions.

C. Frequency of Misconceptions

For each of the nine statements, a frequency table was generated by tallying the number of occurrences of each correct, incomplete, and incorrect transaction in the protocols of the 30 users. Table II lists the key correct transaction(s) and the three most common incorrect or incomplete transactions (i.e., misconceptions) for each of the nine statements.

IV. SUMMARY AND RECOMMENDATIONS

Lack of understanding and user misconceptions can be summarized as follows:

1. *INPUT Statements.* Users have difficulty in conceiving where the to-be-input data comes from (i.e., the keyboard) and how it is stored in memory (i.e., in the indicated memory space). Furthermore, many users fail to understand the nature of executive control, i.e., that the computer will "wait" for input from the keyboard as cued by the question mark on the screen. A major misconception is that "INPUT A" means that the letter A is input and stored in memory. These users need explicit training concerning the role of the input terminal, the wait–run control, and the memory spaces.

2. *READ DATA Statements.* Users have difficulty in conceiving of where the to-be-read data comes from (i.e., the input queue or DATA statement) and how it is stored in memory (i.e., in a specified memory space). For example, a major misconception is that "20 DATA 80, 90, 99" means that the numbers are placed into memory or printed on the screen; another major misconception is that "30 READ A" means to find the value of A and print that value on the screen. Subjects need explicit training concerning the data stack and memory spaces.

3. *Conditional and Simple GOTO Statements.* The major difficulty with the GOTO statement is that users do not understand what will happen after program execution moves on to the desired line. Also, with the conditional GOTO, users have difficulty in understanding what to do if the condition is false. Misconceptions include thinking that "60 GOTO 30" means to find the number 30 (rather than line 30) and "IF A < B GOTO 99" means move to line 99 (without a test). Hence, beginners need training in executive control of the order of execution of statements in a program.

4. *LET Statements.* Users seem to get confused between solving or storing an equation (i.e., treating the equal sign as an equality) and making an assignment. Those who seem to understand the assignment property in the statement will have difficulties in conceiving where to store the assigned value. For example, a major misconception for "LET A = B + 1" or "LET D = 0" is to think that the equation is stored in memory. Beginning programmers need explicit training concerning memory locations and under what conditions values stored in those locations get replaced.

5. *PRINT Statements.* Users seem to confuse the function of PRINT C and PRINT "C", and have difficulty in conceiving that these statements simply display on the screen what is asked to be printed; users incorrectly assume that the computer keeps a record of what is printed somewhere in memory. Beginning programmers need explicit training in comparing the types of PRINT statements.

The present study provides evidence that "hands-on experience" is not sufficient for the productive learning of computer programming by novices. Users tend to develop conceptions of the statements that either fail to include the main idea or that include outright misconceptions. Explicit training is needed including the introduction of a concrete model showing the key locations in the computer (e.g., memory spaces, input stack, etc.), verbal and visual descriptions of the key transactions for each statement, and encouragement of the user to role play "what the computer is doing" for statements and programs. These techniques are reviewed elsewhere [4].

Acknowledgment. The authors wish to thank Jeffrey Marcus and the staff of the Microcomputer Laboratory at the University of California, Santa Barbara for their assistance on this project.

REFERENCES

1. Bayman, P., and Mayer, R.E. Novice users' misconceptions of BASIC programming statements. Rept. 82-1, Series in Learning and Cognition, Dept. Psychology, Univ. California, Santa Barbara, 1982.
2. Marcus, J. *Basic in Six Hours: A Self-Instruction Text.* Microcomputer Laboratory, Univ. California, Santa Barbara, 1980.
3. Mayer, R.E. A psychology of learning BASIC. *Comm. ACM 22*, 11 (November 1979), 589–593.
4. Mayer, R.E. The psychology of how novices learn computer programming. *Comput. Surv. 13*, 1 (March 1981), 121–141.
5. Mayer, R.E., and Bayman, P. Psychology of calculator languages: A framework for describing differences in users' knowledge. *Comm. ACM 24*, 8 (August 1981), 511–520.
6. Moran, T.P. An applied psychology of the user. *Comput. Surv. 13*, 1 (March 1981), 1–11.

CR Categories and Subject Descriptors: K.3.2. (Computers and Education): Computers and Information Science Education—*computer science education, curriculum, BASIC.* D.3.1 (Programming Languages): Formal Definitions and Theory—*semantics, misconceptions.*
General Terms: Experimentation, Human Factors, Languages
Additional Key Words and Phrases: programming, novices, man-machine interface, BASIC.

Received 2/82; revised 9/82; accepted 10/82