Submitted to the Fifth Workshop on Empirical Studies of Programmers
(about 4000 words)

Strategy Choice and Change in PASCAL Programming

Quanfeng Wu & John R. Anderson

Department of Psychology

Carnegie Mellon University

ABSTRACT

Three experiments were conducted to investigate iterative or looping strategy choices and changes, especially between the "**while-do**" and "**repeat-until**" looping constructs, in PASCAL programming. The results from the first experiment, in which subjects were free to choose between the two looping alternatives, indicated that most PASCAL programmers were quite sensitive to problem types and adaptable in choosing appropriate looping strategies. In other two experiments subjects were either forced or primed to use one of the two looping strategies. These experiments revealed that subjects were quite tenacious in using the appropriate strategy and their performance deteriorated when they were forced to use a different strategy. These results are consistent with results of Reder (1987, 1988), Reder and Ritter (1990), and Siegler and Jenkins (1989) on strategy selection in other domains and with other populations.

INTRODUCTION

The issue of cognitive strategy choice and change is increasely attracting attentions of cognitive psychologists. In fact, recently there has been a large body of research conducted to investigate cognitive strategy choice and change in various domains of cognition, ranging from memory retrieval (Reder, 1987; 1988), through solving some laboratory problems such as the Tower of Hanoi puzzle (Simon & Reed, 1976; Simon, 1978; Ruiz, 1989), to doing arithmetic calculations (Reder & Ritter, 1990; Siegler & Shrager, 1984; Siegler & Jenkins, 1989) and performing some other more ecologically valid cognitive tasks such as programming (Gray & Anderson,1987; Katz & Anderson, 1988; Pennington, 1987; Soloway, et al., 1982; Soloway, 1983; Soloway, Bonar, & Ehrlich, 1983; Visser, 1987).

This study is about PASCAL programmers' strategy choice and change in writing iterative or looping programs. Three experiments were conducted. The first experiment adopted a simple design in which subjects were free to choose between the two indefinite looping strategies--namely, the "**while-do**" or the "**repeat-until**" constructs. In the second and the third experiments, subjects were either forced or primed to use one of these two looping alternatives. These three experiments were designed to yield some convergent evidence for testing the hypothesis that most PASCAL programmers are adaptable to problem types while choosing looping strategies.

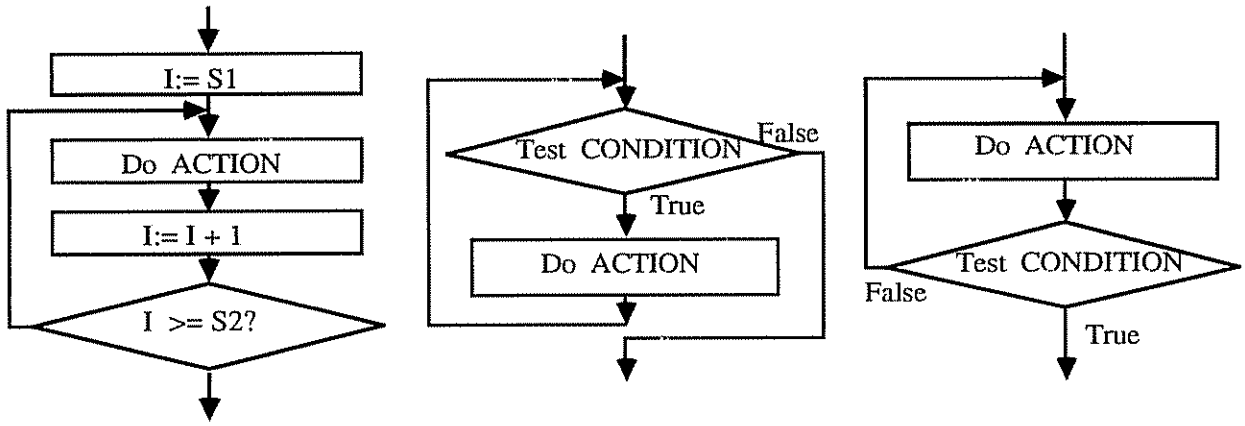
ITERATIVE STRATEGIES IN PASCAL PROGRAMMING

Among computer programming languages, probably PASCAL is the best known conventional or imperative programming language; it is the first advocated and widely accepted structured programming language (Jensen & Wirth, 1974). By now, PASCAL is not only widely used in the computer community but also globally accepted as a tool for teaching structured programming as a good methodology and style of programming. This is the reason why we chose PASCAL as a test-bed for studying looping strategy selection in programming.

With respect to iterative structures, there are three types of iterative (or looping) constructs furnished by PASCAL -- that is, the **for-to-do**, the **while-do**, and the **repeat-until** constructs (or statements). These constructs will be referred as the **F-**, the **W-**, and the **R-**constructs hereafter in the article. The first one is only used in definite looping situations where the number of iterations is known before the iterative part of the program is actually executed. However, the other two may be used in definite as well as indefinite looping cases where the number of iterations may only be dynamically determined by the execution of the looping part. These three types of looping constructs are illustrated by their flowcharts in Figure 1.

The difference between the **F-**construct and the rest is more obvious than the difference between the two indefinite looping constructs. Considering the symmetry of the two indefinite looping constructs, they were chosen to be the subject matter focused on in this research. Note that, for the **W-**construct, the termination condition of iteration is always tested before the execution of the iterative part so that the iteration may be executed zero times (not executed at all). On the

other hand, for the **R**-construct, the test for termination occurs at the exit of the iterative part so that the iteration will be executed at least once. That is the fundamental difference between these two indefinite looping constructs and in fact is the principle typically taught for choosing between them.



Definite looping
for I= S1 to S2
do ACTION
(F-construct)

Indefinite looping

while CONDITION
do ACTION
(W-construct)

repeat ACTION
until CONDITION
(R-construct)

FIGURE 1. Three types of iterative constructs facilitated by PASCAL programming language.

To implement any looping program, either the **W**- or the **R**-construct can be selected and applied. However, in certain cases, choosing the **W**-construct would result in a more concise and well-structured program than it would be if the **R**-construct is chosen; and in some other cases, it is just the other way around. Also there are situations where using either the **W**- or the **R**-construct is neutral. We classified problems into four categories: **W**-problems for which it is easier and more natural to use the **W**-construct; **R**-problems, easier to use the **R**-construct; **NE**-problems, neutrally easy to use either constructs; and **NH**-problems, neutrally difficult to use either constructs. The last class of problems are middle-out looping problems for which the natural place for testing looping termination and quitting loop is in the middle of the iterative body. However, in PASCAL there is no convenient construct for implementing this kind of loop excepting by appealing to the **go-to** statement. We will refer to the strategy to use the **go-to** statement to jump out of the iteration as **G**-strategy. In our experiment, this strategy was only allowed during planning or flow-chart drawing but not in actucally coding in PASCAL. TABLE 1 presents some examples of the four types of problems.

TABLE 1. Examples of four types of problems: W-, R-, NE-, NH-problems.

W-problem (test-before-execution):

Write a program to copy a part of an input file into an output file. The input file is structured as follows:

Name-1	Integer-1
"END"	Integer-(n+1)

Copy the file until the "END" is reached (Including the item 'END').

R-problem (test-after-execution):

Almost the same as the above problem except excluding the item "END".

NE-problem (test-either-before-or-after-execution):

Write a program to read in a series of integers until their sum is greater than 10000 (It is assumed that each integer in the series is less than 10000). This program should also calculate the average of the integers which have been read in.

NH-problem (test-in-middle-of-execution):

Write a program which interactively reads in a month number. That is, the program is expected to get a number in the ranger of 1 through 12. If the value obtained from the input is out of the range, then an appropriate message should be displayed and an additional value should be inputted. This process goes on until a correct month number is obtained, then the correct number is to be displayed.

EXPERIMENT 1: FREE CHOICES OF ITERATIVE STRATEGIES

This experiment was to investigate how PASCAL programmers chose different looping strategies on different types of problems in planning and in programming. Therefore, in this experiment subjects were free to choose among the **W-**, **R-**, and **G**-strategies when planning, and between the **W-** and **R**-strategies while programming.

Method

Subjects. 20 subjects participated in this experiment. They were from CMU (Carnegie Mellon University); among them 2 were undergraduates, 14 were graduates, and 4 were research assistants. All of them knew PASCAL before taking part in the experiment.

Materials. 13 problems were used in this experiment. Among them 4 were classified as **W**-problems, 4 as **R**-problems, 3 as **NE**-problems, and 2 as **NH**-problems (so 5 **N**-problems). The first 9 subjects only solved 8 problems -- namely, 3**NE**+2**NH**+2**W**+1**R** (this is the order in which the problems were presented to the subjects); and the rest 11 subjects completed all the 13 problems -- namely, 3**NE**+2**NH**+4**W**+4**R** (also in the order of presentation).

Procedure. Before the experiment actually began, subjects were asked to fill out a questionnaire form. The form was used to collect some information about the subjects, including

their GRE or SAT scores if known, and their self-ratings of programming experience in different languages. For the self-rating of programming experience, the subjects were asked to rate along a scale from zero to five, for each language which they had mastered, according to their own confidence in that language. There were 11 subjects who filled in GRE math scores and 2 subjects who filled in SAT math scores; and the average math scores across both the 11 GRE and 2 SAT scores are 754. Subject's self-ratings of their experience in PASCAL programming range from 2 to 5, and across all the 20 subjects the average is 3.8.

There were two experimental sessions. In the first session, subjects completed either 7 (for those who finished 13 in total) or 4 problems (for those who finished 8 in total); and in the second session, either 6 or 4. In each session, each subject was first asked to draw flowcharts for all the problems to be solved in that session. Only after the subjects finished drawing flowcharts for all the problems did they begin to write PASCAL programs for these problems. All their flowcharts and the first drafts of their PASCAL programs were worked out on paper. When the first draft of the program was finished, the subjects were required to type in the program to the Macintosh II computer and then use the LightSpeed PASCAL version to test it. They had to debug, if needed, the programs until they yielded the correct results which were required by the experimenter; sometimes, the experimenter provided necessary consultation on PASCAL syntax or about the details of the special PASCAL version. The subjects were not allowed to look at their flowcharts while they were programming; neither were they permitted to access to their solutions on the previous problems when they were on a later problem.

Results

The three looping strategies -- namely, the W-, R-, and G-strategies -- were all exhibited in subject's flowcharts. On the other hand, while programming in PASCAL, subjects were explicitly advised to only use either the W- or R-construct; therefore, only two kinds of looping strategies, corresponding to the two looping constructs, are there manifested in their PASCAL programs. Among the 20 subjects who participated in this experiment, 3 subjects overwhelmingly used the W-strategy in programming (all of them did 13 problems, and on all the 13 problems they used that strategy), and one subject overused the R-strategy (he completed 8 problems; and on 7 of them,

including the 2 W-problems, he used the R-strategy). Here we are only interested in the general pattern of the data from the subjects that varied their use of programming constructs, so these subjects are excluded from the data analysis in this results section. That is, only the data from 16 subjects are analyzed in the following.

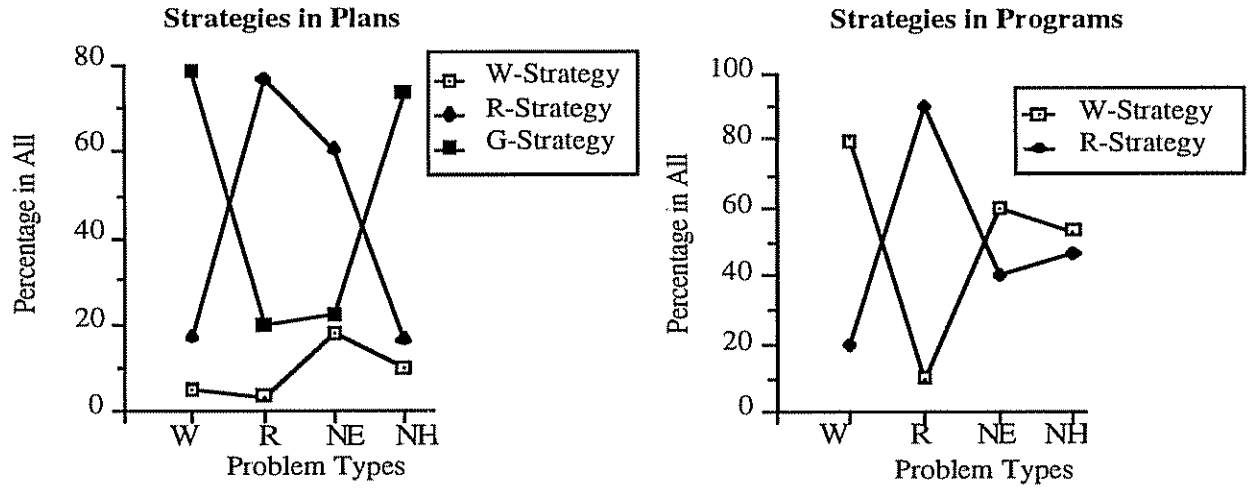


FIGURE 2. The structures (or strategies) manifested in the subject's plans and programs in Experiment 1.

Figure 2 shows the distributions of different strategies chosen by subjects on different categories of problems, classified by their natures, both in planning and in programming. Three separate one-way analyses of variance (ANOVAs) were performed on the data for the three different strategies in planning. The statistical results showed that the effects due to the nature of problem were significant, as for W-strategy: $F(3, 45) = 4.43$, $p < .01$; for R-strategy: $F(3, 45) = 13.06$, $p < 0.001$; and for G-strategy: $F(3, 45) = 30.23$, $p < 0.001$. Another one-way ANOVA was performed on the data for the W-strategy in programming; the results also revealed significant effects due to the nature of problem, $F(3, 45) = 17.96$, $p < 0.001$. The results seen displayed in the figure are as follows: In planning, the W-strategy is a not the preferred strategy for most subjects on any kind of problem; the R-strategy is highly chosen on R- and NE-problems; and the G-strategy is favored on W- and NH-problems. On the other hand, in programming, the W-strategy is dominant for the W-problems; the R-strategy is dominant for the R-problems; however, neither strategy is dominant for the NE- or NH-problems.

By studying the pattern of the strategies chosen by subjects in the planning phase, we can deduce that what are the natural looping strategies on what types of problems, irrespective to any programming language. Therefore, from the above presented results, it seems that the G-strategy is natural on the W- and NH-problems, while the R-strategy is natural on the R- and NE-problems. On the other hand, by studying the pattern of strategies exhibited in programming, we are indeed confirming the hypothesis that most PASCAL programmers are very sensitive to the nature of the problems being programmed and quite adaptable in choosing corresponding and suitable looping strategies in PASCAL iterative programming.

Let's take a close look at the correlation between ability and adaptability. Subject's experience in PASCAL can be given by two measures -- namely, subject's total problem solving times and their self-ratings of PASCAL knowledge. As the result of the data analysis, we see that subject's self-ratings of their PASCAL programming skill are rather subjective and are unrelated to their problem-solving times (the correlation coefficient is -0.058); therefore, we used the time measure as the more objective one between the two measures. The adaptability is defined as the proportion of the number of the W- and R-problems on which subject's chosen strategies are consistent with the natures of the problems to the total number of these two classes of problems. The data analysis revealed that the correlation coefficient between the adaptability and self-rating was 0.137, that between the adaptability and time measure is -0.121, and that between adaptability and GRE math scores is -0.147. Thus, we see that subject's expertise in PASCAL programming does not correlate with their adaptabilities of choosing looping strategies very strongly.

EXPERIMENT 2: FORCED CHOICES OF ITERATIVE STRATEGIES

Following the first experiment, it is reasonable to hypothesize that if PASCAL programmers are forced to use a looping strategy which is incompatible with the nature of the problems for programming then their performance will be hampered in some way. This is essentially the hypothesis the present experiment intended to test.

Method

Subjects. As in Experiment 1, all the 24 subjects involved in this experiment were also from the CMU community, among them 14 undergraduates, 8 graduates, and 2 research assistants. The subjects were reimbursed for their participation in the experiment.

TABLE 2. The design of Experiment 2 on forcing subjects to use predetermined W- or R- looping strategies.

Problems	The sequence of the problems		
	5 N-problems	4 W-problems	4 R-problems
Conditions	Looping constructs forced to use in the experiment		
Group 1	**R**	**R**	**R**
Group 2	**W**	**W**	**W**
Group 3	**Free**	**R**	**W**
Group 4	**Free**	**Free**	**Free**
(Group 4 consists of 8 subjects from Experiment 1)			

Design. There were three conditions in this experiment, with 8 subjects in each condition. The design of the experiment (Table 2) was that subjects were either forced to use the R- (Group 1) or W-strategy (Group 2) on all the problems or forced to use the strategies which were incompatible with the natures of the problems -- namely, to use the W-strategy on R-problems and R-strategy on W-problems (Group 3). Furthermore, we selected the 8 subjects in Experiment 1 who finished all the 13 problems and for whom we knew their GRE/SAT math scores to make up an additional condition (Group 4), that is, a condition in which subjects were free to choose either the W- or R-strategy on all kinds of problems. As seen from Table 2, on the N-problems the first two groups were forced to use the R- and W-strategies respectively, whereas the other two groups were free to use either the R- or W-strategy. According to this design, we would hypothesize that subjects who used the W-strategy on R-problems would spend more programming time than those who used the R-strategy on these R-problems, and that subjects who used the R-strategy on W-problems would spend more time than those who used the W-strategy on these W-problems. We would also hypothesize that on the N-problems these four groups of subjects would not differ very much in their performance. The 24 subjects in this experiment were randomly assigned to the three original experimental conditions; according to their GRE/SAT quantitative scores and self-ratings of their

PASCAL knowledge averaged over the subjects within each group. From the figure we see that subjects in the four conditions are fairly closely matched.

Materials. The same 13 problems as used in Experiment 1were used in the same order in this experiment.

Procedure. The same procedure as used in Experiment 1 was followed here. That is, there were two sessions of experiment: in the first session, each subject completed 7 problems; and in the second session, 6 problems. In each session, the subject first did planning then embarked on actual programming. Most of the subjects were accompanied by the experimenter when they were engaged in experimental sessions, but only for 3 subjects (one in each group) concurrent verbal protocols were collected.

Results and Discussion

To verify the hypothesis mentioned in the design of this experiment, we used the time measure -- namely, the programming time spent by different groups of subjects on different types of problems. The average times of our four groups of subjects are displayed in Figure 7. A two-way ANOVA was performed on the data; the results indicated that the effect due to the nature of problem was statistically significant, $F(2, 14) = 45.29$, $p < .001$; the effect due to the difference among the four groups of subjects was also significant, $F(3, 21) = 23.24$, $p < .001$; and the interaction between them was significant $F(6, 42) = 52.25$, $p < .001$. As seen from the figure, the data indicate that subjects had advantages when they used the strategy which was compatible with the nature of the problems in a class. More specifically, subjects who used the W-strategy on the R-problems spent more time than those who used the R-strategy on these R-problems, and subjects who used the R-strategy on the W-problems spent more time than those who used the W-strategy on these W-problems. When subjects were free to choose they were as fast as subjects who were required to use approriate strategy in solving the problems.

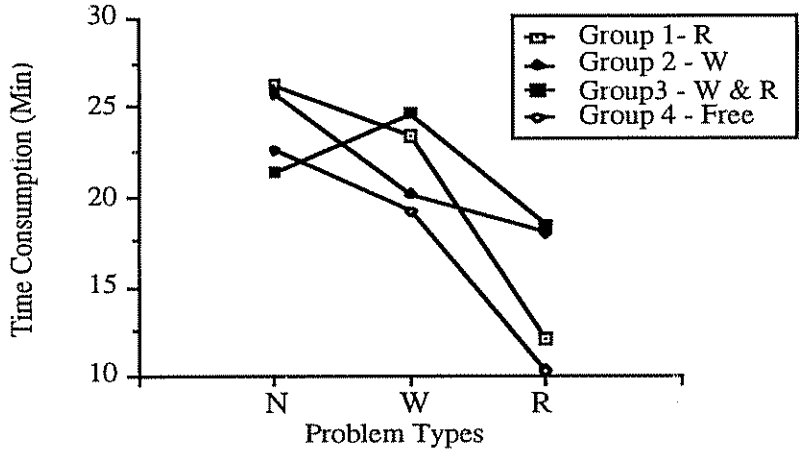


FIGURE 3. The time spent by the subjects in Experiment 2 while they were forced to use predetermined looping constructs with a comparison to the time spent by 8 subjects in Experiment 1 (Group 4) while they were free to choose looping constructs.

EXPERIMENT 3: INDUCED CHOICES OF ITERATIVE STRATEGIES

This experiment was also to provide further evidence supporting our general conclusion on PASCAL programmer's adaptability in choosing looping strategies. We were also interested in relating the adaptability with Einstellung effect in problem solving (Luchins & Luchins, 1959). Specifically, the hypothesis being tested in this experiment was that solving a set of W- or R-problems could have some effect on solving later problems of neutral nature while having less effect on later solving R or W-problems.

Method

Subjects. 32 subjects participated in this experiment. They were either CMU undergraduates (15 of them), graduates (14), or research assistants (3). They were from different departments in CMU and had varied experience in PASCAL programming, and they were randomly assigned to the different conditions of the experimental design.

TABLE 3. The design of Experiment 3 on inducing subjects to use either the W or R looping construct.

Conditions	The sequence of the problems to be presented to subjects
Group W1	**W W W W N N N N N R R R R**
Group R1	**R R R R N N N N N W W W W**
Group W	**W W W W R R R R N N N N N**
Group R	**R R R R W W W W N N N N N**

Design. Four conditions were devised in this experiment (accordingly, four groups of subjects), with 8 subjects in each condition. The different conditions were only different from one another in the order in which the testing problems were presented to the subjects (Table 3). The 32 subjects were randomly assigned to these different conditions of the experimental design. Table 8 provides various descriptive statistics for the four different groups of subjects; there is little differences among the groups.

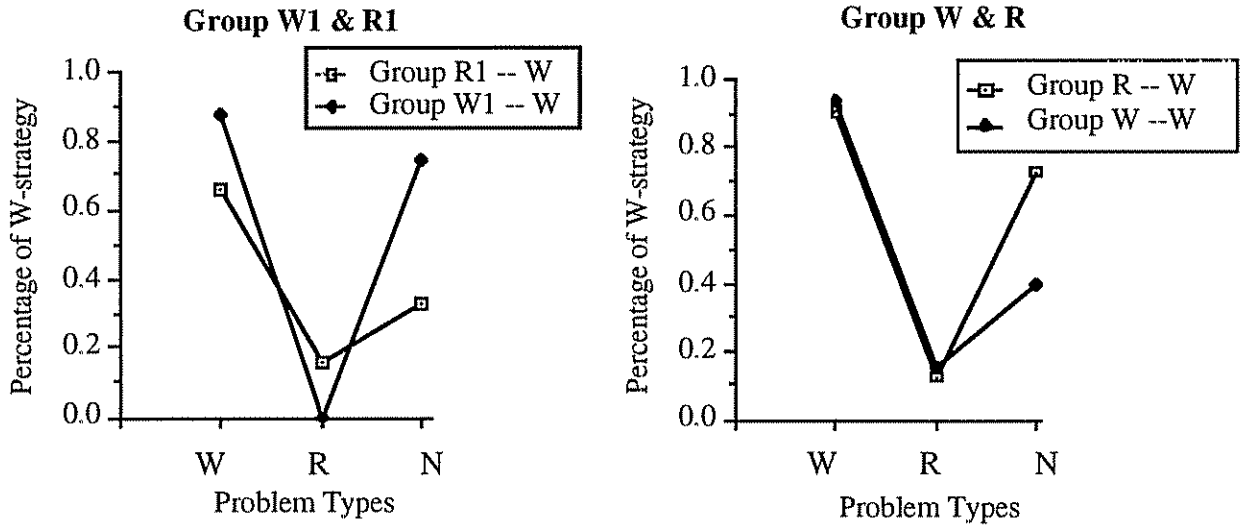
The intention of this experimental design was to see whether there were any inducing or priming effects of having solved a series of problems of one nature upon solving later problems of another type. For instance, for the subjects in Group W1, after they had solved a set of W-problems, they solved a set of N-problems and then a set of R-problems. We were interested in whether any inducing effects would be exhibited on switching from W-problems to N- and R-problems.

Materials. The 13 problems used in this experiment were the same as those used in the previous two experiments. The sequences in which they were presented to subjects were different according to the designated conditions.

Procedure. In this experiment, subjects were only required to work out flowcharts and PASCAL programs on paper in this experiment. The correctness of their programs was judged by the experimenter; if their programs were incorrect then they had to revise them. This procedural change from the previous experiments to the present one was because we here were mainly interested in the looping structures of the programs produced by subjects, not very much in timing subject's problem solving processes.

Results

Again, there were 5 subjects (5/32= 15.6% of the total subject population of this experiment) who idiosyncratically overused either the W- or R-strategy on all the problems. Among these subjects, two used the W-strategy on all 13 problems; one used the R-strategy on all the problems; and the other two used the R-strategy on 12 problems, including on four W-problems. These subjects were excluded from the following data analysis. The data from the experiment expressed as the percentage usage of the W-strategy on the three different types of problems by four groups of subjects are shown in Figure 4. A two-way ANOVA was performed on the data; the results indicated that the effect due to the nature of problem was statistically significant, $F(2, 14) = 170.92$, $p < .001$; the effect due to the different presentation order as for different groups of subjects was marginally significant, $F(3, 21) = 4.20$, $p < .02$; and the interaction between them was significant $F(6, 42) = 4.62$, $p < .001$. As seen from Figure 4, the pattern of the data seems to conform to the hypothesis being tested. That is, on the N-problems, subjects from Group W1, who had used the W-strategy on the previous set of W-problems, tended to use that strategy more often than those from Group R1. The same data pattern holds for Group W and Group R in that the immediately preceding problems biased the strategies they chose on the N-problems, although the disparity between these two groups of subjects on the N-problems is less smaller than that between Group W1 and Group R1. Furthermore, from the data we also can see that there are no significant effects of having previously exposed to a set of R-problems upon solving a set of W-problems later, and vice versa.



NOTE: The problems are not listed in the order of presentation in the experiment.

FIGURE 4. The structures (or strategies) manifested in the subject's programs in Experiment 3 while they were induced to use particluar types of looping constructs.

CONCLUSIONS

With some uncertainty about the range of subjects our conclusions may apply to, we feel that these three experiments have established the following conclusions about strategies for iterative programming in PASCAL: 1. Independent of any particular programming language, there are two types of looping strategies most frequently and naturally chosen by people while planning iterative solutions to the problems being solved. People usually choose the R-strategy if it seems easier to use it, or choose the G-strategy otherwise. 2. In PASCAL programming, most programmers seem to be very sensitive and adaptable to the nature of the program. They would choose between the W- and R-strategies according to which will produce a concise and well structured program. 3. Adaptability in choosing programming strategy does not seem to be related to the programmer's ability. 4. When subjects are forced to use a non-preferred strategy their programming suffers. 5. When the problem does not have a preferred strategy, subjects show tendency to continue to use the last strategy they used.

The high degree of sensitivity displayed by our subjects to problem characteristics is consistent with other results from other domains in cognitive psychology (e.g., Reder 1987, 1988; Reder & Ritter, 1990; Siegler & Shrager, 1984) indicating that subjects are highly adaptive in their

strategy choices. Reder (1987, 1988) showed in a different domain, question-answering, that subject's strategy selection was jointly influenced by the nature of the problem (question) and by what strategies have recently been used. Our results showed both effects; however, unlike her, we see that problem nature totally dominates past experience. This may reflect the fact that the problem nature manipulation in our experiments was stronger than her manipulation which was recency of information. In more recent research on arithmetic problem solving, Reder and Ritter (1990) have also found dominant effects of problem type. Our research is also consistent with the emphasis of Siegler and Jenkins (1989) on the highly adaptive nature of strategy selection. Therefore, the pattern of strategy selection seen in these experiments in consistent with results based on very different populations working in very different domains.

REFERENCES

Gray, W. D. & Anderson, J. R. (1987). Change-episodes in coding: when and how do programmers change their code? in Olson, G. M., et al. (eds), Empirical Studies of Programmers: Second Workshop. New York: Ablex Publishing Corp.

Katz, I. R. & Anderson, J. R. (1988). Debugging: an analysis of bug-location strategies. Human-Computer Interaction, Vol. 3, pp. 351-399.

Luchins, A. S., & Luchins, E. H. (1959). Rigidity of Behavior: A Variational Approach to the Effects of Einstellung. Eugene, OR: University of Oregon Books.

Pennington, N. (1987). Comprehension strategies in Programming. in Olson, G. M., et al (Eds), Empirical Studies of Programmers: Second Workshop, Ablex Publishing Corp.: New York.

Reder, L. M. (1987). Strategy selection in question answering. Cognitive Psychology, 19(1), 90-138.

Reder, L. M. (1988). Strategic Control of Retrieval Strategies. In G. Bower (Ed.), The Psychology of Learning and Motivation, Vol. 21, New York: Academic Press.

Reder, L. M. & Ritter, F. (1990). The effect of feature frequency on feeling of knowing and strategy selection for arithmetic problems.Learning, Memory, and Cognition. (in press)

Ruiz, D. & Newell, A. (1989). Tower-noticing triggers strategy-change in the Tower of Hanoi: a Soar model. The Proceedings of the Eleventh Annual Conference of the Cognitive Science Society. Ann Arbor, Michigan.

Siegler, R. S. & Jenkins, E. A. (1989). How Children Discover New Strategies. Hillsdale, NJ: Erlbaum.

Siegler, R. S. & Shrager, J. (1984). Strategy Choices in addition and subtraction: How do children know what to do? In C. Sophian (Ed.), Origins of Cognitive Skills. Hillsdale, NJ: Erlbaum.

Simon, H. A. & Reed, S. K. (1976). Modeling strategy shifts in a problem-solving task. Cognitive Psychology, Vol. 8, pp. 86-97.

Soloway, E., Bonar, J., & Ehrlich, K. (1983). Cognitive strategies and looping constructs: an empirical study. Communication of ACM. Vol. 26, No. 11 (November, 1983).

Soloway, E. & Ehrlich, K. (1984). Empirical studies of programming knowledge. IEEE Transactions on Software Engineering. Vol. SE-10, No. 5 (September, 1984).