A Quantitative Approach for Choosing a Procedural Programming Language in Freshman Programming

Thomas J. Cortina
Polytechnic University

An important problem that colleges and universities continue to struggle with is the choice of programming language for the first programming course. During the summer of 1996, I ran an experiment to gain quantitative information concerning how well high-school students not familiar with programming adapt to four different procedural programming languages (Pascal, ADA95, C, and C++). The goal of this research is to determine if the choice of introductory programming language has any measurable effect on the learning ability of a typical student in the introductory programming course taught at a typical engineering school. Initial analysis indicates that students learning Pascal or C++ (without the use of objects) for the first time performed marginally better than students that learned ADA95 or C. However, student surveys indicate that several constructs used in ADA95 and C caused these introductory students to have more difficulty. By adjusting the presentation of these topics, the instructor can teach effectively in any of the four programming languages. This paper outlines the design and the results of the experiment, and future work that can extend these results.

1. INTRODUCTION

Colleges and universities continue to struggle with the choice of programming language for the first programming course, typically denoted CS1 [1] [7]. Traditionally, the choice of language has been based on personal preferences of faculty members based on previous experience or influence from industry, high schools, and other faculty members [9]. Many papers have been published describing experiences using one particular programming language to teach CS1 (e.g. [10], [11]).

During the summer of 1996, an experiment was run at Polytechnic University to determine if the choice of programming language truly affects how a student learns how to program a computer. A set of approximately 100 eligible high school students were invited to take our standard introductory programming course, and their performances on assignments and tests were recorded for comparison. Each student was required to have two years of high school mathematics and no prior experience with computer programming (with the exception of some minimal BASIC programming). Students chosen for this course represent typical college freshmen who may pursue undergraduate degrees in science and engineering fields.
Students were separated into four sections randomly, where each section learned procedural programming methodology using a different programming language: Pascal, ADA95, C and C++\(^1\). Algorithms and flowcharts were used in common lectures in order to present the same methodology to all students and avoid bias to one particular language. Computer laboratory sections met separately and reviewed the lecture material using the specific programming language for each section. The chosen textbooks were all written or co-written by the same author in order to minimize the bias toward one particular language.

The goal of this research is to determine if the choice of programming language has any measurable effect on the learning ability of a typical engineering student in an introductory programming course. Analysis of the programming styles and examination results indicate that students in the Pascal and C++ sections performed marginally better than students in the Ada95 and C sections. Student surveys given at the end of the class indicate that several topics in the Ada95 and C sections caused additional programming troubles not experienced by students in the Pascal and C++ sections. If the instructor presents the topics differently, it appears that any of the four languages is suitable for beginning programming students.

Section 2 of this paper describes the format for the experiment including the course curriculum, the selection of students and the placement of students into course sections. Section 3 describes the method used for quizzesing students during the semester along with results for each of the quizzes. Section 4 continues with a discussion of the final exam, including its design and a comparison of the results of this exam to students in an equivalent programming class at Polytechnic University. Section 5 compares the final grades of the students to those of typical programming classes given in the past three years at Polytechnic University using Pascal and C++. Section 6 outlines the results of student surveys concerning their work in the course. Section 7 analyzes the results of this experiment and provides outlines of future experiments in progress or in development to extend these results.

2. EXPERIMENT DESIGN

Students were selected for this experiment in the summer of 1996 from local high schools in New York City and Long Island through the Center for Youth in Engineering and Science (YES) at Polytechnic University. High school students selected met the following criteria:

- Completion of Level 2 of the integrated mathematics curriculum of New York State (basic knowledge of algebra and trigonometry).
- An overall average in mathematics and science courses of at least 80.
- No prior experience with high-level computer programming (with the exception of BASIC programming).
- Availability to attend all class meetings with no more than 2 absences.

---

\(^1\) Only procedural aspects of C++ were presented in the C++ section. Additionally, students used input and output streams (\texttt{cin} and \texttt{cout} instead of \texttt{scanf} and \texttt{printf}). Students in this section did not learn object-oriented programming.
As an incentive to encourage students to apply for this experiment, students who completed this course successfully earned 3 credits which could be used at Polytechnic University or transferred to another institution upon advisor approval.

Classes met for four days a week for 90 minutes each day over a period of eight weeks for a total of 48 meeting hours, compared to our regular freshman class with a total meeting time of 39 hours with an additional 13 optional hours of reserved computer lab time. Students were taught by the same instructor (the author of this paper) in order to minimize bias due to the presentation style of an individual instructor compared to another. For each week of classes, students met together in a lecture room two days a week for a common lecture, and then separately in the computer laboratory based on their assigned sections, as shown in Figure 1.

Students were assigned to sections A, B, C or D randomly. The method used was to take the complete alphabetized student roster and assign the first student to section A, second to B, third to C, fourth to D, fifth to A, etc. As students were assigned to a particular section, no mention was made with regards to the language that the section would use. At the end of the first class, the students were informed of the specific language they would study in each section. Approximately 15 students requested section changes. The majority of these students wanted to be moved to section D (C++) due to the current popularity of that language. Students were moved between sections only on documented reasons including summer employment conflicts and transportation difficulties. The final registration totals for the four sections is given in Figure 2. It appears that four students were moved from section B to section D, but a total of 10 students were moved between sections, with the final result shown in Figure 2.

<table>
<thead>
<tr>
<th>SECTION</th>
<th>LECTURE (Mondays and Wednesdays)</th>
<th>COMPUTER LAB (Tuesdays and Thursdays)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A (Pascal)</td>
<td>10:30AM-12:00PM</td>
<td>9:00-10:30AM</td>
</tr>
<tr>
<td>B (ADA95)</td>
<td>10:30AM-12:00PM</td>
<td>10:30AM-12:00PM</td>
</tr>
<tr>
<td>C (C)</td>
<td>10:30AM-12:00PM</td>
<td>2:00-3:30PM</td>
</tr>
<tr>
<td>D (C++)</td>
<td>10:30AM-12:00PM</td>
<td>3:30-5:00PM</td>
</tr>
</tbody>
</table>

**Figure 1:** Scheduled class hours for the four sections based on programming language.

Students were requested to purchase the textbook for their assigned sections. In order to minimize the bias introduced by using textbooks by different authors with different presentation styles, the textbooks were selected to have similar style and quality.

<table>
<thead>
<tr>
<th>SECTION</th>
<th>NUMBER OF STUDENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>A (Pascal)</td>
<td>25</td>
</tr>
<tr>
<td>B (ADA95)</td>
<td>21</td>
</tr>
<tr>
<td>C (C)</td>
<td>25</td>
</tr>
<tr>
<td>D (C++)</td>
<td>29</td>
</tr>
<tr>
<td>TOTAL</td>
<td>100</td>
</tr>
</tbody>
</table>

**Figure 2:** Total registration for all sections based on programming language.
styles and different subject ordering, books either written or co-written by the same author were used for this course [3], [4], [5], [6]. In general, the presentation of the material in these four texts was similar, including programming examples and chapter exercises. In a few cases, the examples in the textbooks were not similar, and handouts were distributed to help each of the classes with class material not in its textbook.

The course curriculum for each section was identical and is given in Figure 3. Students were given a curriculum sheet that listed the course curriculum and the pages to read for their particular textbook. With the exception of software engineering and ethics, all course topics were covered in each of the four textbooks in significant detail; therefore, students had enough reading material to study from regardless of the programming language class to which they were assigned.

<table>
<thead>
<tr>
<th>WEEK</th>
<th>TOPIC(S)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction to Computers, Elementary Programming Concepts</td>
</tr>
<tr>
<td>2</td>
<td>Conditional Programming</td>
</tr>
<tr>
<td>3</td>
<td>Programming with Loops</td>
</tr>
<tr>
<td>4</td>
<td>Subroutines, Scope of Variables</td>
</tr>
<tr>
<td>5</td>
<td>Data Types, Programming with Files</td>
</tr>
<tr>
<td>6</td>
<td>Arrays, Intro. to Algorithm Analysis using Elementary Sorting and Searching</td>
</tr>
<tr>
<td>7</td>
<td>Elementary Data Structures (Multidimensional Arrays, Strings, Records)</td>
</tr>
<tr>
<td>8</td>
<td>Software Engineering and Computer Ethics, FINAL EXAM</td>
</tr>
</tbody>
</table>

Figure 3: Course curriculum for all four sections of the experimental course.

Seven quizzes were given to students, once a week beginning on the second week, covering material from the previous week. In addition, there were seven programming homework assignments for the first seven weeks and a student survey given as homework during the eighth week. Programming assignments were run under a simple UNIX environment with standard UNIX compilers for all four programming languages. Results from the quizzes (35%), exam (45%), and homework (20%) make up the student’s final grade. Analysis of the quizzes, exam, and survey are given in subsequent sections of this paper.

3. QUIZ DESIGN AND RESULTS

---

2 Results from the programming assignments are not presented. Programs were written to monitor how many times a student compiled each programming assignment, but system problems caused many of these data files to be corrupted throughout the summer. Therefore, these sketchy results are not included in this paper.
A quiz was given to all students during the first lecture class of each week, starting with the second week of class. Each quiz consisted of 10 multiple choice questions worth 5 points each and one programming question worth 50 points. The questions on a given quiz were all identical, with the exception that students in each section saw code written in their given programming language.

For each of the multiple choice questions, the respective answers were also identical except for the programming language used. As an example, the four versions of a typical question from the Quiz 3 (programming with loops) is given in Figure 4. All quizzes were formatted in the same manner in order to ensure that all classes received the same questions using the same presentation. For each quiz, students received 5 points for each correct multiple choice answer for a maximum of 50 points.

Figure 5 summarizes the results of all 7 quizzes on the multiple choice questions for each class. It can be seen that students in the C and ADA95 sections performed below average on a majority of the questions, scoring significantly below the class average in many cases.

For the programming part of each quiz, students received 50 points for a correct program with a maximum of two minor syntax errors (e.g. a missing semicolon, using a bracket instead of a parenthesis, etc.). A maximum of 10 points were subtracted for syntax errors, a maximum of 30 points were subtracted for errors in program logic and algorithm design, and a maximum of 50 points were subtracted for a completely incorrect answer. Programming problems were designed to be language neutral and easily solvable in any of the four languages so as not to present any bias toward one programming language.

Figure 6 summarizes the results on the multiple choice questions and programming problem of each quiz for each class. It can be seen that the students in the Pascal and C++ sections have performed better on each of the quizzes, in some cases earning average grades 10% higher than those students in the ADA95 and C sections. Additionally, by comparing the performance of the classes on the programming problems, it can be seen that students in the C++ sections averaged approximately a 40% score, which indicates that a majority of the students in this class made simple syntax errors in general as opposed to major algorithm or logical errors. On the other hand, in classes like ADA95 and C, students had more difficulty with the program problems, earning sometimes less than an average 30% score on the programming problem. This indicates that these students had more difficulty with the algorithm and logic details of the program in addition to the significant syntax errors.

PASCAL VERSION
What is the output of the following code fragment?

```pascal
x := 0;
for I := 1 to 5 do
  for J := 1 to I do
    x := x + 1;
write(x);
```

(a) 5  (b) 10  (c) 15  (d) 25  (e) none of the above
ADA95 VERSION
What is the output of the following code fragment?
\[
x := 0;
\text{FOR I IN 1..5 LOOP}
\quad \text{FOR J IN 1..I LOOP}
\quad \quad x := x + 1;
\quad \text{END LOOP;}
\text{END LOOP;}
\text{Ada.Integer_Text_IO.Put(Item=>x);}
\]
(a) 5   (b) 10  (c) 15  (d) 25  (e) none of the above

C VERSION
What is the output of the following code fragment?
\[
x = 0;
\text{for (I=1; I<=5; I++)}
\quad \text{for (J=1; J<=I; J++)}
\quad x = x + 1;
\text{printf("%d", x);}\]
(a) 5   (b) 10  (c) 15  (d) 25  (e) none of the above

C++ VERSION
What is the output of the following code fragment?
\[
x = 0;
\text{for (I=1; I<=5; I++)}
\quad \text{for (J=1; J<=I; J++)}
\quad x = x + 1;
\text{cout << x;}
\]
(a) 5   (b) 10  (c) 15  (d) 25  (e) none of the above

Figure 4: Four versions of the same quiz question given to students during QUIZ 3.
Figure 5: Average amount of correct responses on each quiz question for each programming class.
Figure 6: Average total score on each quiz for all four programming classes, including average score on multiple choice questions and average score on programming questions.

4. FINAL EXAM DESIGN AND RESULTS

The three-hour final exam of this class was designed to be similar in length to the standard Advanced Placement examinations in Computer Science[2]. The multiple choice section of the final exam contained 40 questions, and the programming section contained 4 programming problems. In order to determine how well the classes understood the material with respect to one another, variations on multiple choice questions from the quizzes were chosen for the final exam. Figure 7 gives an example of a variation of the question shown earlier in Figure 4 which was also included on the final exam.
PASCAL VERSION
What is the value of x after the following code is run?
\[
x := 0;
\text{for } W := 1 \text{ to } 5 \text{ do }
\quad \text{for } Y := W \text{ to } 5 \text{ do }
\quad \quad x := x + 1;
\]
(a) 5 (b) 10 (c) 15 (d) 25 (e) none of the above

ADA95 VERSION
What is the value of x after the following code is run?
\[
x := 0;
\text{FOR } W \text{ IN } 1..5 \text{ LOOP }
\quad \text{FOR } Y \text{ IN } W..5 \text{ LOOP }
\quad \quad x := x + 1;
\quad \text{END LOOP;}
\text{END LOOP;}
\]
(a) 5 (b) 10 (c) 15 (d) 25 (e) none of the above

C VERSION
What is the value of x after the following code is run?
\[
x = 0;
\text{for } (W =1; W<=5; W++)
\quad \text{for } (Y=W; Y<=5; Y++)
\quad \quad x = x + 1;
\]
(a) 5 (b) 10 (c) 15 (d) 25 (e) none of the above

C++ VERSION
What is the value of x after the following code is run?
\[
x = 0;
\text{for } (W =1; W<=5; W++)
\quad \text{for } (Y=W; Y<=5; Y++)
\quad \quad x = x + 1;
\]
(a) 5 (b) 10 (c) 15 (d) 25 (e) none of the above

Figure 7: Four versions of the same final exam question. This question was included since the average correct response to the original version of this question (Figure 4) was 31%.

Of the 70 quiz questions given during the summer semester, 38 questions were chosen to be included on the final exam. All of these questions received an average correct response of less than 70%. Variations of each multiple choice question were designed with either a permutation of the same answer set or new answer sets with similar features. An additional two new questions were designed to cover software engineering and computer ethics principles that were not covered on any previous quiz. It is important to note that students were not informed of the design of the final exam (i.e. the repetition of questions from quizzes). Therefore, students were not inclined to study specific questions from the previous quizzes in order to do better on the final exam.
Figure 8 gives the average results of the multiple choice questions of the final exam for each of the four programming sections along with the class average. It is interesting to note that there is significant improvement in nearly all of the final exam questions, especially for the ADA95 and C classes, yet the C and ADA95 classes still performed below average on a majority of the questions.

![Figure 8: Performance on final exam multiple-choice questions for all four classes.](image)

Figure 9 shows a comparison of the average performance of each question with the average performance of the original quiz question on which the exam question was based. The overall performance improved for nearly all questions. Some notable occurrences are outlined below.

![Figure 9: Average percentage of multiple choice questions answered correctly on the final exam compared to the original versions given on previous quizzes.](image)

Questions 6, 7 and 8 were a combined question concerning the following code fragment shown in Pascal (assuming a value for x is input from the program’s user):

```pascal
y = 1;
if x <= 0 then y := y + 1;
if x >= 2 then y := y + 2;
writeln(y);
```

Students were asked to determine what value for y cannot be output from this program, how many test cases were needed to test this program fragment, and what if/else fragment performs
the same operation as the code shown. Although the first version of this three-part question in the second quiz was a disaster for all classes, all classes showed significant improvement in understanding this construct.

Question 10 illustrates a loop with an apparent compound statement for its body (note the missing brackets) which students failed to recognize the first time in the original quiz question as shown below in C or C++:

```c
for (j=1; j<=4; j++)
    if (j == 1) x = 1;
    else x = x + j;
cout << x;
```

It is interesting to note that the “trick” question above cannot be written in ADA95 which must have an END LOOP statement in the proper place, yet many ADA95 students still got the problem wrong in the original quiz version (Quiz 3, Question 2) since the same “false” indentation was used.

Questions 12 and 13 asked for the value of y after the following loop executes as shown in ADA95 and also with the statements in the loop body interchanged:

```ada
x := 0; y := 1;
WHILE (x < 4) LOOP
    x := x + 1;
    y := y * x;
END LOOP;
Ada.Integer_Text_IO.Put(Item => y);
```

Although students perform much better in determining the output of the loop with the statements in the loop body interchanged (Question 13), students actually performed worse determining the output of the original loop (Question 12).

Question 33 asked how many elements were in an array based on its variable declaration. The original quiz question (Quiz 7, Question 1) showed poor performance for the C section despite the fact that both C and C++ sections were taught that subscripts for arrays begin with 0. The results in the final exam show that the C students did much better with that concept. Finally, question 38 covers text files and had a decrease in the final exam, and questions 39 and 40 represent the questions on software engineering and ethics which were not covered in any quiz.

The multiple choice final exam questions for the C++ class were also administered to the freshmen programming classes at Polytechnic University in the Fall 1996 semester as a homework assignment, and their results compared to the summer classes is shown in Figure 10. Although the freshmen students did not solve these questions in an examination environment, students were instructed to work alone, use only 90 minutes, and students were given credit for this assignment for simply handing in the work. Since the freshmen students knew they weren’t being graded on the actual answers, this minimized the amount of cheating and group
collaboration. The results here are shown simply for a general comparison with typical engineering students who learned the same material in one chosen language (C++). A noteworthy observation is that even though the freshmen had 13 weeks to cover the same material instead of 8 weeks and the freshmen were not in an examination environment when the questions were completed, the freshmen performed poorly on most questions in comparison with the high school students in the experiment. A possible explanation may be due to the fact that the high school students had seen variations of the final exam problems on earlier quizzes whereas the freshmen students had not.

![Figure 10: Performance on final exam multiple choice questions for college freshmen (using C++) from Fall 1996 and all high school students in the summer program during Summer 1996.](image)

5. FINAL GRADES AND COMPARISONS WITH TYPICAL PROGRAMMING CLASSES

Final grades for students in this experiment were assigned using the same grading scheme that is used in standard freshman programming classes at Polytechnic under my supervision. The grading key is shown in Figure 11. Figure 12 displays the percentage of students earning each grade for each of the experimental classes. In addition, the figure also displays how the average grade assigned compares with other freshman programming classes at Polytechnic University.

<table>
<thead>
<tr>
<th>AVERAGE</th>
<th>GRADE</th>
<th>AVERAGE</th>
<th>GRADE</th>
</tr>
</thead>
<tbody>
<tr>
<td>90-100</td>
<td>A</td>
<td>74-76</td>
<td>C+</td>
</tr>
<tr>
<td>87-89</td>
<td>A-</td>
<td>70-73</td>
<td>C</td>
</tr>
<tr>
<td>84-86</td>
<td>B+</td>
<td>67-69</td>
<td>C-</td>
</tr>
<tr>
<td>80-83</td>
<td>B</td>
<td>64-66</td>
<td>D+</td>
</tr>
<tr>
<td>77-79</td>
<td>B-</td>
<td>60-63</td>
<td>D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0-59</td>
<td>F</td>
</tr>
</tbody>
</table>

![Figure 11: Grading key for final grades for students in all sections.](image)
It can be seen clearly from Figure 12 that the Pascal and C++ sections had more A and B grades given than the ADA95 and C sections. More than 50% of the students in the Pascal and C++ sections earned grades of A or B. Also, the C section had a significantly high number of failures, much greater than any of the other sections. Section 7 of this paper combines these results with the student survey responses to suggest possible reasons for the given student performances.

6. STUDENT SURVEY RESULTS

During the eighth week, students were given a survey to complete that asked questions concerning their progress throughout the course. The number of survey responses was 85%, since this was a required homework for all students (students received a grade of 100 for this
homework if the survey was handed in, regardless of the answers given). The total number of respondents to this survey was 86 out of 100 (Pascal 24, ADA95 21, C 20, and C++ 21). A summary of the major survey questions is given below.

Students were asked about how extensive their computer background was before starting this course. The results are shown in Figure 13. It is clear that most students have used computers before, yet less than half had programmed in BASIC programming (usually as part of a high school course in computer fundamentals).

Additionally, students were asked to indicate which topic was the most difficult of the eight units covered in the course. The results of this question are shown in Figure 14. Clearly, students found subroutines (including parameter passing) and elementary data structures (e.g. records or structures) difficult. Interestingly, students did not report finding conditional (“if”) statements difficult, even though performance on that section of the course was not outstanding (see Section 3).

The student body in this experiment were very science and engineering oriented, as shown in Figure 15. Many of these students expressed a serious interest in applying to an engineering or
science school, and therefore these students represent typical students that would take freshman programming courses in many engineering and science oriented university and college programs.

![Figure 15: Survey responses indicating students desired college major.](image)

At the beginning of the course, a number of students wanted to switch to other sections after they learned which language they would be studying (see Section 2). One of the survey questions asked the students if they wanted to learn a different language (now that they’ve learned one already). The responses are shown in Figure 16. Approximately half of the Pascal students were satisfied with the language assignment, while the other half desired another language (mainly C++). The ADA95 students were more displeased with their language assignment; many students in this section described the other languages as more popular and wished to study those instead. The C students who wanted to study another language overwhelming wanted C++, which is interesting since the C++ students learned nearly the same material and syntax (with the exception of parameter passing and input/output handling). Clearly, the students in the C++ section were very satisfied with their language assignment with the exception of one student who desired ADA, running counter to the overwhelming trend for students to use the popular C++ language.

![Figure 16: Survey responses to a question asking whether students wanted to study a language different than the language to which they were assigned.](image)
The final question students were asked concerned an issue that many engineering and science colleges face given their choice of programming language for the introductory programming course. Quite often, the language used in the introductory course is used in subsequent advanced courses, such as courses in data structures, algorithm design, and software engineering. Students were asked if they would continue into the next programming course (modeled after the CS2 curriculum [1], [8]) even if the course used a procedural language different than the one they learned. The survey responses are shown in Figure 17. A number of students indicated “maybe”, which means that school constraints would make it difficult or students felt that the new language would make the second course much more difficult to learn. The fact that few students answered a definitive “yes” indicates that there is concern over what computer language they are exposed to as they enter college. A conflict exists between the programming language taught in high school and the language chosen by universities as their main programming language. Some universities even use non-procedural paradigms (e.g. Scheme) to teach freshman students. [9] Students will be reluctant to learn languages in high school that they cannot use directly in college, even though many instructors agree that the concepts of procedural programming can be taught with any of these languages.

7. CONCLUSIONS AND FUTURE WORK

Based on quiz and exam results and the survey responses, there are several topics that cause all students difficulty, regardless of the programming language that is used. Students have difficulty understanding the concept of parameter passing, regardless of the syntax used for each language. In addition, students had difficulty understanding how loops work; this is evident due to the fact that variations of nine of the ten quiz questions on loops were included on the final exam due to their low success rate.

One factor accounted for a large number of the difficulties experienced by the ADA95 section. The textbook [3] presents packages in early chapters, which is meant to encourage good software design. However, the long package names caused many students in this class to generate excessive syntax errors simply due to typographic errors. In addition, on quizzes, students in the
ADA95 section consistently finished their quizzes last due to the excessive writing involved when including package names. It is interesting to note that when students were shown how to bypass the repeated use of the package name after the fifth week of class, many ADA95 students immediately began using it to shorten the amount of time to write their code and had much less difficulty getting programs to compile on the computer.

Several factors accounted for a large number of difficulties experienced by the C section. Language features like call by reference using pointers caused a great deal of confusion not experienced by other sections, including the C++ section. Students leaving out an ampersand or asterisk while writing functions would get errors that were difficult to trace. As a result, a higher percentage of students in this section did not complete program homework assignments successfully or on time. An additional factor that may have influenced the results of this section was the dynamic of the class itself. Whereas the other three sections were very responsive during class and learned to work together in the lab very quickly, the students in the C section were reluctant to participate in general and found it difficult to work in groups. It is not clear if this behavior is due to the choice of programming language (and its inherent difficulties) or the combined personalities of the students randomly placed in this section.

Future work in this area is necessary in order to determine how well students respond to a particular programming language. Additional summer classes are scheduled at Polytechnic University to repeat this experiment with additional qualified high school students. In these experiments, the textbook and the computer environment will be altered to study if these elements make a significant difference in the student performance for any particular language. Also, other procedural languages like FORTRAN90 and Modula-3 may be considered for evaluation. Additional research will be performed to determine how to analyze the results for any statistical significance. Finally, the author is studying how effective alternative paradigms (e.g. functional, logical, object-oriented, etc.) may be as beginning programming languages.

Instructors who wish to use the quizzes, examinations and assignments of this experimental course as the basis of their own experiment are encouraged to do so. All quizzes, examinations, and course information are available through the World Wide Web at the Web address http://cis.poly.edu/cortina/research.htm. Instructors may wish to vary the textbook, presentation of materials or programming environment used in order to determine if these changes play a significant role in the advantage of one language over any other language.

REFERENCES


**Thomas J. Cortina** is an instructor of computer and information science at Polytechnic University. His research interests include computer science education, programming languages and computer music. He is currently completing his PhD at Polytechnic University in the area of the analysis of programming language instruction on college-level students. Cortina received a BS and MS in computer science from Polytechnic University.