https://doi.org/10.48009/3_iis_2018_1-10
Issues in Information Systems
Volume 19, Issue 3, pp. 1-10, 2018

TEACHING MULTIPLE LANGUAGES CONCURRENTLY

Jeanne M Baugh, Robert Morris University, baugh@rmu.edu

ABSTRACT

This paper investigates the issue of teaching multiple undergraduate introductory programming languages during the same semester. The languages taught were C, Java, COBOL and Visual Basic. The same large programming project was assigned to each class. An analysis was done as to which language proved to be the most challenging while teaching the project requirements. Additionally, student's perceptions of the project is also discussed. This project model introduces the programming concepts at critical times during the duration of the project. The systematic process of creating a well-designed project leads to a significant coding effort on the part of the first time programmer. However, depending on the language that is being taught, variations in the amount of time spent on a specific task as well as when the task is introduced is a requirement.

Keywords Project based learning, Beginning Programming, Language constructs

INTRODUCTION

Coding efforts are often large projects worked on by many individuals and comprising many, many lines of code. But, how does one give the very beginning programming student the experience of writing and documenting a large program, when they don't even have any idea of the concept of a variable? There are many building blocks of logic and syntax that must be mastered before a large program can even be attempted. Among these are variables, control statements, loops, sub-programs and arrays.

According to Wang, programming requires thinking with abstract concepts, which is difficult for novices. Secondly, programming includes many different tasks, such as problem solving, algorithm and data structure design, programming language comprehension, testing, and debugging (Wang, 2010). The student who has never programmed before feels an enormous sense of accomplishment when the code all comes together and produces correct results.

There are many different approaches to teaching programming. This is evidenced in the number of programming textbooks available to an Instructor. Two Instructors teaching the same course at the same University may be offering a completely different version of the same course material. Each Instructor has his own methods of teaching and the academic freedom enjoyed in the classroom is something that is not taken lightly (Hamilton, 2007**)**. But what does an instructor do if they are teaching more than one language during the same semester? What if they are teaching four different languages during the same semester? Although many will agree that each programming language contains the same basic logic and syntax building blocks, some languages are harder that others, not only for the student to learn, but also for the Instructor to teach! (Laurent, Utkeb, 2016)

The Accreditation Board for Engineering and Technology (ABET) (ttp://www.abet.org) dictate that for an Information Systems Degree the student must take two semesters of a programming language. Often, the student will be required to complete a large programming effort in the second course, but such a project is also doable in a very beginning course, if it is structured well. This author has had a great deal of success implementing a large programming project in C++ and Java. The method used is to introduce topics as they are needed in the project. Organizing the presentation of the topics in C++ and Java during the same semester has not been a significant problem for this author, in that the "guts" of each language is somewhat similar. But as many know, languages such an COBOL, C, Java and Visual Basic contain very different structures to implement the same logical tasks.

This author was asked to teach four different programming languages at two institutions during the same semester. This was not planned by the author, but it was seen as an opportunity to compare teaching each of the languages using the same project. This paper describes the method used by the author, to teach beginning programming in COBOL, C, Java and Visual Basic during the same semester. The same large programming project was assigned to each set of

programing students. Project-based learning increases long-term retention of content, motivates the students to do additional work, illustrates to the students the value of the materials covered, and most importantly, provides practical experiences that enrich the student's academic experiences. (Albanese & Mitchell, 1993; Hutchings & Wutzdorff, 1988; Strobel & van Barneveld, 2009; Walker & Leary, 2009)
The paper also discusses how each programming language implements the necessary tasks and in the opinion of the author, how difficult each of those implementations is to teach.

COURSE STRUCTURE

Researchers have investigated project-based learning in a wide variety of disciplines and settings and have generally found it to be effective in increasing student motivation and improving student problem solving and higher order thinking skills, addressing different learning styles, and providing students with an integrated learning situation. Because the eventual outcome of the course is a significant programming project (approximately 1000 lines of code) it was essential to have a very structured approach to the course work (Baugh, Kovacs 2012). This structured approach applied not only to what had to be taught, but when it was taught.

Problem/Project based learning has shown to produce positive student outcomes. Although Savery suggested that the problem/project should be freeform (Savery 2005), this author created a structure for the project, but also allowed for creativity on the part of the programmer. There are often many ways to accomplish the same outcome in a specific language and the students were given freedom to do just that.

Figure 1. indicates the basic progression of the programming topics in each course that eventually lead to a large programming effort.

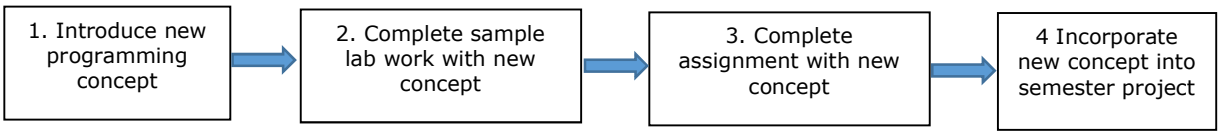

Figure 1. The basic progression of the programming topics in each course

It has always been the practice of the author to provide a great deal of sample code for the students, no matter what programming language was being taught. Short sample programs seem to be a good way to illustrate a specific programming concept. Sample code is an essential part of all four programming courses described in this paper. As previous research as shown, this had proven to be extremely helpful to programming students (Baugh, Kovacs, 2012).

Basic Assignments
Frist, in each of the four languages, a number of small programming assignments (usually 4 to 5) were designed to allow the student to master some of the basic building blocks of each language. The building blocks were necessary before the large programming project could be started. A summary of the major topics covered within these assignment is as follows:

1. Numeric variables
2. String variables
3. Input and output statements
4. Simple assignment statements
5. Math expressions
6. Conditional statements
7. Loops
8. Subprograms

After the first eight basic topics were taught in the various languages, the project was begun. This came at about midterm on the semester. The sequence of assignments and concepts were then as follows:

- Phase I assigned – This phase covered the completion of the entire structure of the menu driven programming project using stub methods/ functions/ subprograms/ paragraphs
- The concepts of array definition, parallel arrays and manipulation of arrays were taught
- File definition, contents and reading of text files
- Phase II assigned – Reading of data from various files into three sets of parallel arrays and reports generated from the arrays
- Phase III assigned– Modifying the parallel array data (add, delete and modify) and writing the data back to the data files

Analysis of concepts per Programming Language

1. Numeric variables

C, Java and Visual Basic all provide a straightforward method of defining these data types. The author stressed the idea of "do you need decimal points or not". The lecture for this topic in these three languages did not vary too much.

However, in COBOL numeric variable definition is defined in working storage using picture clauses and is not quite as simple as other languages. Because there are many different permutations of picture clauses, students can have a great deal of difficulty with this topic. At this point of the semester, the author did not require students to be proficient with all variations of picture clauses.

2. String variables

Defining strings in COBOL, Java and Visual Basic was not seen as being very difficult to convey to the students by the author. Understanding that a string is basically anything that you can not do math with was the method used to stress such storage. However, in the C language there is no string variable and is defined as an array of characters. Trying to teach arrays during a first week of class was just not realistic. Therefore the topic of strings was put off until much later in the semester.

3. Input and output statements

Input and output with COBOL is simple using the display and accept statement, but this topic presented challenges in each of the other three language. Teaching Dialog boxes in Java and text boxes in Visual Basic was similar. The concept of reading everything in as text and converting it, made input and output tougher in these languages. But teaching reading and writing data in C was much more difficult. Getting the students to understand formatting factors and memory addresses is very difficult topic at the beginning of such a course.

4. Simple assignment statements

The instruction of assignment statements in all of the four languages was not seen as difficult, although COBOL uses a much different syntax. The concept of: do what is on the right of the equal sign, come up with one value and finally store it in the left, translates within all languages.

5. Math expressions

Precedence of operations is basically the same in all languages. But integer math is always difficult to convey to students. Additionally, COBOL's math syntax is much more difficult for the students to understand versus a simple equal sign. (The COMPUTE statement is available)

6. Conditional statements

Conditional statements (if statements) are always difficult because there are so many ways they can be written. An instructor must struggle to provide as many examples as possible. In reflection, the author could find no difference in the difficulty of teaching if statements among the languages. It was tough in all of the four languages.

7. loops

In C, Java, and Visual Basic, the syntax was very similar, so the instruction was not difficult. COBOL syntax is much different and required more effort to teach on the author's part. Although loops are very often a very difficult topic to teach in any language.

8. Subprograms

Subprograms is an area that is difficult in almost any language. However, COBOL seemed to be the easiest to teach. Presenting the PERFORM statement to students helped in understanding the concept of subprograms. In fact the PERFORM statement was taught very early in the course. Of all the concepts taught, the author feels that teaching sub programing in C, Java and Visual Basic required a great deal of effort beyond all other concepts.

9. Arrays

Defining arrays in C, Java, and Visual Basic are very much the same from an instruction point of view. Defining arrays in COBOL is somewhat different in that they are called tables, but really did not require more effort from the author's point of view.

10. Files

All of the four languages presented challenges to the author in teaching the concept of reading data from text files. There seemed to be no easy language here when it can to working with text files.

SEMESTER PROJECT

Having a large programming project in a first programming course is difficult to accomplish. As stated earlier, one must be organized. However, assigning the same large semester project in four different programming languages presents more challenges. To keep organized, the project was the same for all of the four languages:

- Read data from three different text files into three different sets of parallel arrays
- Add, delete and modify data in the arrays
- Run various reports based on the data in the arrays
- Write the arrays back to the three data files when the program terminates

Therefore, keeping all project tasks straight was very easy. (The project description for the Java class is provided in the appendix)

The author teaches programming by providing a great deal of sample code for the students. In this case where four different sets of sample code was needed, the author would write the code in one language and then convert the code into the other three languages. Being organized in this manner made it much easier to keep everything straight for each course.

COURSE RESULTS

Again, this author has been teaching beginning programming as previously described in this paper for many years. Those students who were able to complete most of the functionality of the programming project were given an "A" for the course grade. Even though a grade is assigned, what is really being assessed is the level at which the student programs at the end of the course. Therefore, the actually exams given during the term were weighted at a much lower value. About 70% of all students received an A for their specific course.

Because this author has used the programming project as the assessment tool for many semesters, a critical look at the project is always taken. For example, a number of semesters ago, reading data from files was moved to an earlier phase of the project because it was noted that the students seemed to understand that concept at an earlier stage.

As stated earlier, a few things were taught out of sync in each class. For example, in the C course, strings were put off much later in the semester that when they were introduced in the other three programming courses. Since a string in C is a defined as an array of characters, strings were not taught until arrays were first introduced.

Figure 2 summarizes the difficulty level assessed by the author when teaching the various programming courses. A value of 1 to 5 was assigned to each programming concept in each of the four languages. (1 being the least difficult to 5 being the most difficult).

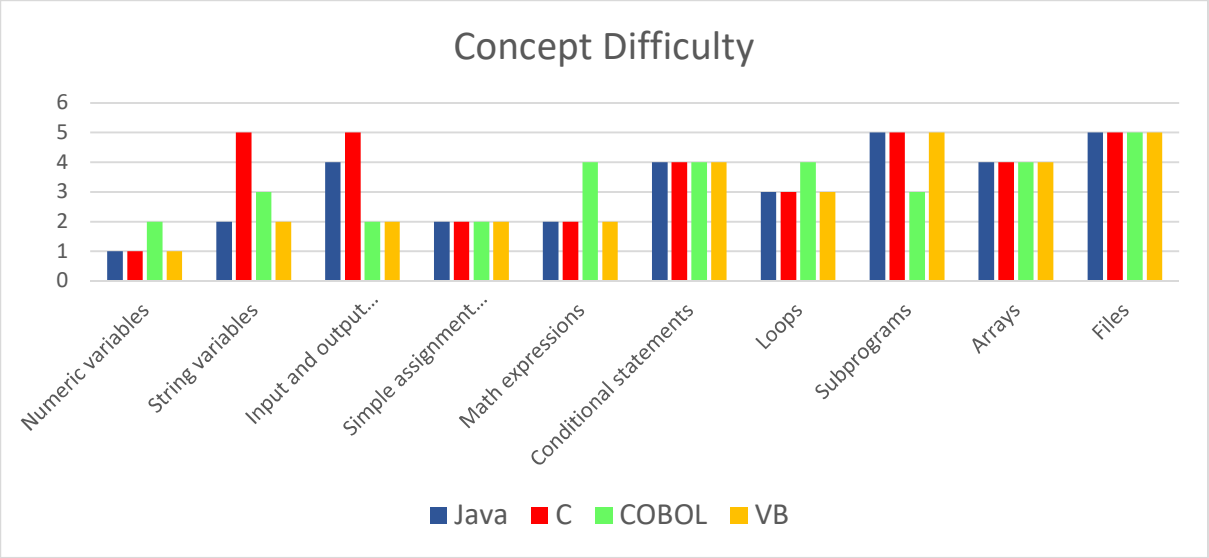

Figure 2. Complexity of concepts in various languages

The total for each language was calculated in Figure 3. As can be seen, this author felt that C was the hardest of the four languages to teach. In reviewing the results, the author feels that the concepts of input/output and strings lead to this conclusion. The feedback of the students supported the author's view.

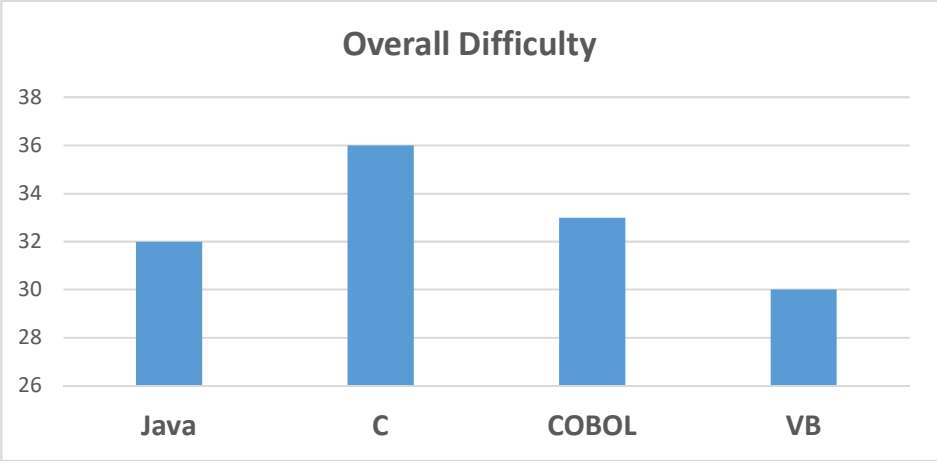

Figure 3. Summary Data

CONCLUSIONS AND RECOMMENDATIONS

There are many right ways to teach beginning programming. The method described here has been extremely successful, considering that most of the students were first time programmers. Students were also very proud of their efforts. Watching the transformation of the students from not understanding a simple output statement to writing a large project encompassing many subprograms is quite rewarding to the author.

This author has been teaching beginning programming, using a large project such as described in this paper for many years. As seen in the literature, this approach has given the students a significant head start in becoming a programmer. Real world software comprises multiple sections of code written by multiple programmers. This project simulates that experience on a smaller scale, of course. But most first-time programmers do not often experience a large project, mainly because there are many programming tasks they must first master.

Those teaching a course such as this must be very organized and attentive to the students. A policy of answering the student's email within 24 hours is extremely important. Students need to feel that the instructor is accessible. A study done by Yang and Cornelius also supports the concept that the students must receive feedback on a timely basis (Yang, Coprnelius, 2004). Help was available and given when needed. One student said; "Without a doubt this was one of the best classes I have ever taken. The teacher was readily available; she responded to my emails fast and was very nice."

Often the programming books are difficult for the student to read, in that a specific programming concept may be embedded within a complicated example. Therefore, the sample code proved to be a valuable resource to the students. However creating four sets of sample code can be very time consuming. And, a few times the author accidently passed out Java code in a C class (the "guts" of the code is so similar!)

Cheating is an issue for programming courses and it is almost impossible to prevent it completely. But having the students explain their code on a traditional exam could help in this area. If the student does not do their own programming, it will be impossible for them to thoroughly explain what they turned in as their own work.

In terms of assessment, this author thinks that learning to program is the most important outcome of any programming course. There should be a continuous improvement process by any instructor of any programming course taught, to make sure that the goals of the course are met.

It is not the recommendation of the author to teach four different programming languages during the same semester. But, if organized, it can be successfully done. Watching students from four different languages move from not knowing what a variable is to writing code that is approximately 1000 lines long, was very rewarding for the author. It was also rewarding for the students. They were very proud of their work.

It must be noted that future research will be undertaken to assess what concepts students in the various languages feel are the most difficult. But from the author's observation, students seemed to concur with the findings in the paper.

As always, not all students will succeed with the project, no matter which language they are using. However, for the majority of the students, they possessed a true dedication to getting all of the functionality of the project to work correctly.
The experience of teaching four different programming courses at the same time, while difficult, was also extremely rewarding to the author.

REFERENCES

Albanese, M., & Mitchell, S. (1993). Problem-based learning: a review of literature on its outcomes and implementation issues. *Academic Medicine, 68*(1), 52-81.

Baugh, J. M., Kovacs, P. (2012). Large Programming Projects for the beginning programmer. *Issues in Information Systems, 13*(1), 85-93.

Hamilton, N., (2007). Faculty Autonomy and Obligation, *Academe, 93*(1), 36-42, Retrieved from:http://www.abet.org/accreditation/accreditation-criteria/criteria-for-accrediting-computing-programs-2016-2017/

Hutchings, P., & Wutzdorff, A. (1988). Experimental learning across the curriculum: assumptions and principals, *New Directions for Teaching and Learning*, *35*, 5-19.

Laurent, H. Utkeb, J. (2016). Programming language features, usage patterns, and the efficiency of generated adjoint code. *Optimization Methods & Software, 31*(5), 885–903.

Savery, J. R. (2006). An overview of PBL: Definitions and distinctions. *The Interdisciplinary Journal of Problem-based Learning, 1*(1), 9-20.

Strobel, J., & van Barneveld, A. (2009). When is PBL more effective? A meta-synthesis of meta-analyses comparing PBL to conventional classrooms. *The Interdisciplinary Journal of Problem-Based Learning, 3*(1).

Walker, A. & Leary, H. (2009). A problem-based learning meta analysis: Differences across problem types, implementation types, disciplines, and assessment levels. *Interdisciplinary Journal of Problem-based Learning, 3*(1), 12-43.

Wang, X. (2010). Teaching programming skills through learner-centered technical reviews for novice programmers. *Software Quality Professional, 13*(1), 22-28.

Yang, Y, C., (2004). Students' Perception towards the Quality of Online Education: A Qualitative Approach. Association for Educational Communications and Technology, 27th, Chicago, Il. October 19-23, 17.

APPENDIX

Sample Java Project: JAVA Project – Phase I
In Phase I, you will be creating the entire structure of a program that will be completed in pieces (phases) In this phase of the project, you are to make calls to methods, but the methods will have only a System.out.println statement in them. This is called a stub. The program will call the proper method and will print a statement, but the rest of the code for the each method will be completed in later phases.
You are to create a program that will keep track of information for RMU. The program will be menu driven and is to look something like the following:

College Registration System
1. Add/modify Student information
2. Add/modify Course information
3. Register Students in courses
4. Report Section
5. Exit RMU System

Please make your selection >

In this first phase you are to write the complete the method for the **menu** and **stubs for all of the other methods.**

- The names and definitions of the methods are as follows:
  1. **Menu** – method that will print the above menu and will return the user's selection to the main program
  2. **Modify_student**– stub- to be completed in a later phase
  3. **Modify_course**– stub- to be completed in a later phase
  4. **Modify_register-** stub -to be completed in a later phase
  5. **Report** – stub- to be completed in a later phase

You must use the above names for the methods in your program. Each method, except the menu method will just be a stub and when called, will print a message to the user that the method has been executed. Control will then pass back to the menu. You will need a while loop to keep the program running until the user picks "5. Exit RMU System "

- You are also to create three other methods that will **run only once** at the beginning of your program.
  6. **Read_student– stub- to be completed in a later phase**
  7. **Read_course – stub- to be completed in a later phase**
  8. **Read_register – stub- to be completed in a later phase**

- Create another method that will run only once at the end of your program.
  9. **end_program– stub- to be completed in a later phase**

Sample Java Project: Phase II
You are to use the code from Phase I and add to it. You are to add code to the following methods:

✓ **Read_student**
- This method will be the first thing to run when your program executes (before the while loop). This method will open a text file called **studentj.dat** and read student information into parallel arrays. This data file is on blackboard and must be copied into the main folder of your JAVA project. The student information that will be read in is as follows:
  - **Student number (a five digit integer code)**
  - **Student last name (string)**
  - **Student first name (string)**
  - **Student major (string)**
  - **Student_gpa (double)**

✓ **Read_course**
- This method will open a file called **coursej.dat** and read information into a second set of parallel arrays. This method also must be executed before the while loop. This data file is on blackboard and must be copied into the main folder of your JAVA project. The course information that will be read in is as follows:
  - **course number (string)**

- **course name (string)**
- **days (string -MWF or TTH)**
- **time (string –xx:xxam or xx:xxpm)**
- **Professor last name (string)**

✓ **Read_register**

- This method will open a file called **registerj.dat** and read information into a third set of parallel arrays. This method also must be executed before the while loop. This data file is on blackboard and must be copied into the main folder of your JAVA project. The registration information that will be read in is as follows:
  - **Registration student number (5 digit integer)**
  - **Registration course number (string)**

✓ **Report**

When this method is called, the following menu will print:

RMU Report Menu

1. Student Master List
2. Course Master List
3. Register Master List
4. List of Students by Major
5. List of Courses by Professor
6. Honor Students
7. Student Schedule for one Student
8. Course Roster for one Course
9. Course Roster for all Courses
10. Report 10
11. Exit Report Menu

Please make your selection >

The above menu will keep running until the user selects **11. Exit Report Menu**

Each report is defined as follows:

1. Student Master List – Displays all data read in from the student file
2. Course Master List –Display all data read in from the course file
3. Register Master List – Displays all data read in from the register file
4. List of Students by Major – Prompt for the major and print the students from that specific major
5. List of Courses by Professor –Prompt for the Professor name and print the courses for that specific professor

Reports 6 thru 10 will be defined in Phase III of the project

You are to upload a word document to blackboard that contains the following:

- **Source code for the program**
- **Screen shot of each report running on the screen**

Sample Java Project: Phase III

You are to add code to your Phase II code to do the following:

1. Modify_student– Add code to this method as follows:

In this method you are to ask the use what they want to do: (something like the following):

Modify Student Information

1. Add Student
2. Delete Student

- When the user picks menu option #1, prompt the user for a new student and store it in the student parallel arrays
- When the user picks menu option #2, prompt the user for the student number, search the student arrays for the student and delete the student from the arrays

3. Modify_course -Add code to this method as follows:

In this method you are to ask the use what they want to do: (something like the following):

Modify Course Information

3. Add Course

4. Delete Course

- When the user picks menu option #1, prompt the user for a new course and store it in the course parallel arrays
- When the user picks menu option #2, prompt the user for the course number, search the course arrays for the course and delete the course from the arrays

4. Modify_register -Add code to this method as follows:
In this method you are to ONLY add new registration information

5. Exit_program-Add code to this method as follows:
This method will write all data from the three sets of parallel arrays back to the **studentj.dat, coursej.dat** and **registerj.dat** files. Therefore, if the user entered new data or deleted data, the current arrays would be over-written back to the data files. Then, the next time the program would run, that new data would be read in.

6. Report -Add code to this method as follows:
In this method, you are to finish the reports.

- **Report #6 – Honor Students** – List of all student information for students with a GPA >= 3.50
- **Report #7 – Student Schedule –** Prompt for the student number and print the student number and name, then print all course information for the courses that student is registered for
- **Report #8 – Course Roster –** Prompt for the course number and print all information for that course. Then print the list of students who are enrolled in that course
- **Report #9** – **Course Roster for all courses** - for all courses, print all information for the course. Then print the list of students who are enrolled in that course
- **Report #10** – This report is of your choice