Teaching Tip

Using the Same Problem with Different Techniques in Programming Assignments

Michael Newby
Department of Information Systems & Decision Sciences
California State University, Fullerton
Fullerton, California 92834, USA
mnewby@fullerton.edu

ThuyUyen Nguyen
Teesside Business School
University of Teesside
Middlesbrough, TS1 3BA, United Kingdom
T.Nguyen@tees.ac.uk

ABSTRACT

Programming assignments are used to assess a student's understanding of the theoretical aspect of programming and their ability to put that theory into practice. When assigning programs for students to complete, it is necessary to make sure that the problem is well specified, realistic, yet is able to be completed in a relatively short period of time. In addition, each assignment should require the use of a different technique. Developing new problems for each assignment is not only time consuming for the instructor, it also requires the student to understand the problem before they can start to write the program using the specified technique. Whilst this is not a bad thing, it sometimes means that students do not really know why they use a particular technique, apart from it being part of the requirements. In this paper, we describe an approach that uses the same problem for all programming assignments within a course. The only difference between the assignments is the technique to be used. This allows students to compare techniques, see the advantages and disadvantages of them, and improve their programming style.

Keywords: Effective Instruction, Programming Style, Programming Assignments

1. INTRODUCTION

There are a number of approaches to teaching programming (Fincher, 1999), and all of them stress the importance of adopting a good program style, as well as producing programs that are correct, robust, and easy to modify. Unfortunately, many students, as well as some instructors, believe that programming is easy. They believe that as long as the program works, they have achieved their goal. However, many other instructors, including ourselves, believe that good style makes it easier to create programs that are correct, and also easy to modify. McAndrews (2000, p.6) points out that students do whatever it takes to get a program to work and do not worry about quality, discipline or planning. He also states that the practices used in college are often carried over to their jobs. So the importance of teaching good programming style from the beginning cannot be overstated. Another point is that while students are 'being shaped' to learn a good programming style, they need to learn and understand the subject matter well. Standler (2006) states that "education is about learning to think", and, in our view, nowhere is this more important than when students are learning programming skills.

In our experience, we have found that when teaching students who have little or have no background in programming style, one of the most important things to do is to encourage critical thinking throughout their learning process. At the same time, we try to discourage them from adopting 'bad' habits, which usually arise from trying to make a program work at all costs.

One approach to teaching programming is to use the Applied Apprenticeship Approach (Astrachan and Reed, 1995), in which students are given examples of 'good' code and are then are given programming assignments, which they are expected to solve using similar techniques. One drawback we have found with this approach is that students often do not know why they use one particular technique rather than another. Frequently, they cannot see the advantages of functions and classes over using in-line code, since they find it quicker to write the in-line code. This should come as no surprise as nearly all programming assignments in most courses require program development, and functions and classes only come into their own when we write a program with the expectation that it will be modified.

We have adopted a variation of the Applied Apprenticeship Approach by using the same original problem with different techniques for their programming assignments so that they can see the differences at each level, but the functionality and, hence, the results are the same. In other words, we are adopting an "experience approach" because if the students already know about the "problem", their main focus then will be on the programming technique to be used instead of trying to understand a new problem together with applying a technique they have just learned. A similar approach has been taken by Ritzhaupt and Zucker (2006) to teach Object-Oriented programming in the Visual Basic.NET environment, but the requirements of the problem chosen for their programming exercises are extended at each level. The approach described in this paper uses the same problem, and can easily be adapted to any programming environment.

2. THE EXPERIENCE TECHNIQUE STRATEGIES

The approach we are using to help student enhance their programming skills involves using the same problem, so once the students understand the problem, they can focus on the programming. In our sixteen-week semester course, a student must learn the basics of programming (in either C++ or Java), the use of methods, the use of classes, and inheritance. At the end of the course, they should also know the advantages of the different approaches. During the semester, students must complete four programming assignments which require at least one two-way statement (an 'if' statement), a case statement, and a loop, in each program. If the application involves a graphical interface, the loop may be unnecessary. These are minimum requirements and the problem can be as simple or as complex as desired. Input data should be validated in the program. In general, the students will be required to write the program using:

1. Direct in-line with no modularity (first assignment)
2. Modularity using functions or methods (second assignment)
3. Modularity using a class (third assignment)
4. Inheritance (final assignment).

After a programming assignment has been graded, a preferred solution is posted and that allows students to see how theirs could be improved. They are also allowed to use any code from one assignment in the following one.

3. TECHNIQUE LEVELS

In this paper, we are using an example of a travel company that provides tour packages for its customers to three regions Europe, Asia, and Australia. To make it easier for customers to make their choice on which package to buy for their trip, the company has adopted a simple pricing formula to calculate the basic cost, discount and airfare. These costs depend on travel region, and the number of days in advance that the package is booked. The specification of the problem is given in the Appendix.

3.1 Direct In-line Code

In the first assignment, students are to learn about the problem and understand its logic. They are to use in-line code to solve the problem. This will involve repeated code for each region, since the default option (which represents invalid input) will differ from all other options. There are ways of overcoming the repeated code (such as setting a flag to indicate that the data is invalid), but these are not satisfactory and are often clumsy. Other problems can occur from the depth of nesting statements leading to a program that is difficult to follow. The only advantage of this approach is that students can write the program quickly, without the need to think about functions or classes. The disadvantages are many, the major one being that every time we add another tour region, then the new code will be copied into the program, modified and the physical size of the program will increase.

3.2 Functional Modularity

In assignment two, students are required to write a solution for the same problem using functions. Using functions or methods to write the program solution leads to a simpler main program. Methods to validate input would mean that the data values processed in the main program are valid and hence there would be no default option in a case statement leading to the reduction in repeated code. Methods would be used for calculations and the main program becomes relatively simple, making it much easier to understand. In the example given in the Appendix, in addition to the methods for validation, there would be methods for:

Basic land-only tour cost
Discount amount
Cost of the airfare
Tour region as a String
Airfare included as a String

There could also be methods for the other calculations such as:

Cost of land-only tour after discount
Cost of trip

In the preferred solution, the main program consists of a set of method calls, perhaps inside a loop or in a method that handles a graphical interface event. The only drawback that may arise in using this approach is the number of parameters that are passed from method to method, and the need to call the methods in a particular order. The first problem may be overcome by using global variables, but that is something that we believe should be discouraged as it can often lead to poor design.

3.3 Class Modularity

The third assignment involves writing a class that contains description of the data and the methods for the calculations to solve the problem. In the example, the class would be a tour, and would include all the data needed to identify the tour package and customer, and the methods needed for the calculations. Validation methods would not be part of the class, since the data should be valid before the object's variables are given those values. Since most of the data used in the calculations is stored within an object of the class, there are fewer parameters, so that using the methods is simplified. Also, the methods may be called in any order, since the main program does not need to know how the methods are evaluated.

The 'type' of object that the instance represents would be stored in an instance variable, and this variable would be used as the case variable in many of the methods. To add a new case would require modifying the methods in the class, but not the main program. In the main program, data would be input and validated, an object of the class would be constructed, and then the object's methods are called to perform the calculations.

3.4 Using Inheritance

The use of inheritance simplifies the code in main even further. The superclass contains the data and the methods that every object has in common, with variations being placed in the subclasses. The type of the object is no longer stored as an instance variable, but the object is an instance of the subclass. In the example, the superclass would be a tour, with subclasses for Asia, Europe, and Australia. With inheritance, case statements will not normally be used within the classes, and the only case statement would be in main when the actual object of the appropriate subclass is constructed. After that, polymorphism takes care of selecting the appropriate methods for the object with dynamic casting only being needed if the class of the object contains a method not specified in the superclass. To add a new type of object requires a new subclass and a change to the case statement (usually in the main program) that constructs the object. Variations in the methods for the different subclasses are easily demonstrated.

4. CONCLUSION

In a sixteen week course, students have to learn and master the subject matter from basic skills in the language to the use of inheritance. For most students, this is a challenge and many others get lost along the way. Using a different problem for each technique would mean that for each assignment, the students would have to familiarize themselves with the problem background to have a better understanding of what is to be done. Using the same problem for all assignments means that students can focus their efforts on how to apply the new techniques to a problem with which they are already familiar. The main benefit of this approach is that students can compare the different solutions to the same problem, and from that see the purpose and advantages of methods, classes and inheritance. Another advantage is that, in order to get a good grade for the program, it must not only 'work', but must also use the correct technique. It overcomes the frequent complaint from students who ask why they did not get an A for a working program. We adopted this approach five years ago, and prior to that, used different problems to assess competency in each technique. Although the evidence is anecdotal, we believe that students who took the course in the past five years have a better understanding of both how and why we use methods, classes and inheritance.

5. REFERENCES

Astrachan, O. and Reed, D., (1995). "AAA and CS 1: The Applied Apprenticeship Approach to CS 1." Proceedings, 26th SIGCSE Technical Symposium on Computer Science Education, Nashville, Tennessee, pp. 1-5.

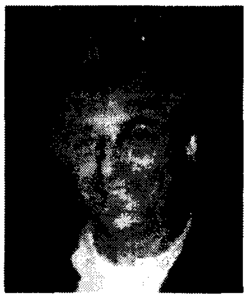
Fincher, S., (1999). "What are we doing when we teach programming?" Proceedings, 29th ASEE/IEEE Frontiers in Education Conference, San Juan, Puerto Rico, pp. 12a41-5.

McAndrews, D. R. (2000). "The Team Software Process (TSP): An Overview and Preliminary Results of Using Disciplined Practices." Technical Report CMU/SEI-2000-TR-015, ESC-TR-2000-105. Carnegie-Mellon University, Software Engineering Institute.

Riezhaupt, A. D. and Zucker, R. J. (2006). "Teaching object-oriented programming concepts using Visual Basic.NET." Journal of Information Systems Education, Vol. 17, No. 2, pp. 163 – 170.

Standler, R., (2006). Standler's Teaching Style. Retrieved January 5, 2007 from http://www.rbs0.com/teaching.htm

AUTHOR BIOGRAPHIES



Michael Newby is a lecturer in the Department of Information Systems & Decision Sciences at California State University, Fullerton (CSUF). He received his B.Sc. from the University of London, U.K, his M.Sc. from the University of Bradford, U.K. and his Ph.D. from Curtin University in Western Australia. In both 2002 and 2005, he received an award as Outstanding Teacher-Scholar at CSUF. His research interests include computer learning environments, and improving student learning in programming courses.

ThuyUyen Nguyen is a Ph.D. candidate at the University of Teesside, Middlesbrough, United Kingdom. She received her B.A. in Business Administration and her M.S. in Information Systems at California State University, Fullerton, where she was an adjunct faculty in the Department of Information Systems & Decision Sciences in 2006. Her research interests include Customer Relationship Management and technology innovation in Small-Medium-sized Enterprises.

APPENDIX

Typical Problem Specification:

A travel company provides tour packages for its customers to three regions Europe, Asia, and Australia. To make it easier for customers to make their choice on which package to buy for their trip, the company has adopted a simple pricing formula. You are to write a program to calculate the cost of each trip, which depends on choices that each customer buys. For this program you must use classes.

The basic land-only tour costs for different regions are:

Europe	$950.00
Asia	$1250.00
Australia	$1550.00

The price discount based on advance booking is:

90 days or more	15%
between 30 and 90 days	5%
less than 30 days	0%

The airfare is charged separately from the land-only tour. For those customers who wish to include the airfare in the package, it is calculated as followed:

To Europe:	60% of the land-only tour price
To Asia:	75% of the land-only tour price
To Australia:	120% of the land-only tour price

The airfares are not discountable. For each customer, you are to input the following:

- Booking Number
- Customer Name
- Travel Region (as a character)
- Number of Days book in advance (as an integer)
- Whether airfare is included (as a character 'Y' or 'N')

The tour region is to be represented as a single character as follows:

Europe	'E'
Asia	'A'
Australia	'U'

After doing so, you are then to output for each customer the following values:

- Booking Number
- Customer Name
- Tour region (as a String)
- Price of land tour-only before discount
- Discount amount
- Cost of land-only tour after discount (Total cost of land-only tour before discount – Discount amount)
- Airfare included (a string that says whether airfare is included or not)
- Cost of airfare
- Cost of the trip (Cost of land-only tour + Cost of the Airfare)

Customer's information should be processed repeatedly until the null string is input for the Booking Number, at which point the following totals should be displayed:

- Number of customers
- Total Discount for all customers
- Total Cost for all customers

The program should then terminate.

Information Systems & Computing
Academic Professionals

STATEMENT OF PEER REVIEW INTEGRITY

All papers published in the Journal of Information Systems Education have undergone rigorous peer review. This includes an initial editor screening and double-blind refereeing by three or more expert referees.

ISSN 1055-3096