How to Produce Students Who Can Solve Problems using Computers Instead of Computers that Create Problems for Students in Engineering

Darrin M. Hanna, Jerry E. Marsh, and Richard E. Haskell
Oakland University
Rochester, MI 48309
Email: dmhanna@oakland.edu

Abstract

Most undergraduate curricula in engineering require engineering majors to take an introductory programming course. Languages such as Fortran, Pascal, and BASIC were used in the past to introduce algorithmic thinking and general programming. More recently, universities have adopted C++ or Java as the new language of choice taught in these introductory courses. This paper presents a course using Visual Basic as the programming language to learn event-driven programming with a graphical user interface, basic constructs of programming, algorithmic thinking, and interfacing with a database while designing software in the laboratory to solve problems in engineering and computer science. For non-computer science students, a course of this type teaches skills necessary to solve engineering problems in their field using a simple language. For computer science students, this course prepares them to take the next course using a language such as C++ or Java where more attention can be paid to the powerful language and object-oriented programming instead of basic procedural constructs. This course is a 4-credit course with a 2-hour lab and 1-hour problem-solving session.

Introduction

Engineering majors typically take an introductory programming course in languages such as Fortran, Pascal, BASIC, and more recently, C++ or Java. These languages represent current programming environments and offer computer science and computer engineering majors a strong foundation in programming knowledge. Developing basic skills for algorithmic thinking while developing an understanding of an object-oriented language is difficult for some students, especially students not majoring in computer science or computer engineering. Obtaining programming skills that will become a tool for solving problems is complicated because of the rigor of learning algorithmic thinking, object-oriented programming, and syntax using a language as complex as C++ or Java. Understanding a programming language is only valuable if the students also develop the ability to apply the knowledge to solve problems they will

encounter in their field by writing a computer program. Presently, many engineering students learn from these courses that they really don't ever want to program again.

This paper presents a course using Visual Basic as the programming language to learn event-driven programming with a graphical user interface, basic constructs of programming, algorithmic thinking, and interfacing with a database while designing software in the laboratory to solve problems in engineering and computer science. The problem solving topics presented in class include statistics, electric circuits, projectile motion, Riemann sums and errors, linear interpolation, statistical data mining, and data normalization. These topics form the basis for problems given to the students to solve using Visual Basic while they are learning to program. For non-computer science students, a course of this type teaches skills necessary to solve engineering problems in their field using a simple language. For computer science students, this course prepares them to take the next course using a language such as C++ or Java where more attention can be paid to the powerful language and object-oriented programming instead of basic procedural constructs. This course is a 4-credit course with a 2-hour lab and 1-hour problem-solving session. Taught in this manner, the class can be offered to large numbers of students by having the entire class attend a single large lecture. In addition, students attend a problem-solving session with one-quarter of the students in each section, and a lab with approximately twenty students per section. Instead of running three smaller sections of a programming course, universities can designate the three faculty positions to this single course, one for the lecture, one to conduct the problem-solving sessions, and one to work in the labs with the lab instructors. This way, students work with faculty in all aspects of the course instead of the traditional faculty lecture and graduate student assisted lab with no additional overhead to the university faculty.

Course Objectives

A successful student in this course will be able to:

- Solve fundamental problems in Engineering and Computer Science.
- Use events in the design and implementation of graphical user interfaces.
- Use forms, buttons, textboxes, radio buttons, and list boxes in Visual Basic.
- Develop Visual Basic code for functions, loops, and decision structures (if, case).
- Use the Visual Basic debugger to watch variables and program execution.
- Implement simple data structures using variables and arrays.
- Interface to sequential files and a database using Visual Basic.

The Lecture

All students attend a common lecture. Since this is a 4-credit course, there are approximately three and a half to four hours of lecture each week. During the Winter 2001 term at Oakland University, there were 175 students registered for this course. A typical lecture accomplishes one or more of the following objectives:

1. Algorithmic thinking and programming conventions, Visual Basic syntax, and Visual Basic controls and events
2. Problem solving topics
3. Laboratory exercises

In the lecture students are taught how to develop algorithms, how to program in Visual Basic, and how to use Visual Basic controls. Students are taught how to take a problem and break it down into a sequence of instructions using a flow chart. The course introduces Visual Basic syntax including conditional statements, loops, functions, one- and multi-dimensional arrays, file input and output, and designing a simple database to use with Visual Basic. The controls covered in the course include the textbox, command button, label, frame, radio button, check box, combo box, list box, timer, line, Microsoft Chart, slider, and Microsoft Comm controls. Basic events are presented such as the Lost Focus, Got Focus, Change, Click, and Timer events. It is important that the instructor not teach the course as a Visual Basic syntax course; instead, Visual Basic must be presented as a tool with which to solve problems instead of detailing the language. In other words, topics are presented for students to be able to use Visual Basic for solving problems and not for students to learn every aspect of the language. For example, arrays are taught as zero-based arrays with length of *max+1*. We do not introduce the concept of using different lower bounds for the array even though Visual Basic easily supports it. This pedagogy makes it possible to introduce in a single semester how to develop an algorithm, syntax for the tool including using databases, and problem solving topics. Additionally, most languages, such as C++, use zero-based arrays. Teaching the course in this manner allows students easily to extend the experience to other languages in the computer science curriculum. Students are encouraged to learn more about Visual Basic on their own.

Problem solving topics are also introduced in the lecture. Typically, an introduction to the topic is given with an example or two. The topic is expanded upon in the problem solving sessions and additional examples are given leading to a homework assignment that students complete on their own.

Each laboratory exercise is introduced in the lecture. The laboratory exercises drive the lecture topics. In order to present the next lab assignment, the problem-solving topic must be introduced along with the controls, events, and Visual Basic syntax necessary to complete the lab successfully.

Problem Solving

Problem solving is the cornerstone of the approach for teaching this course. Since the students are engineering and computer science students, using problems in engineering and computer science as a basis for developing computer programs prepares them for using Visual Basic to solve problems in future courses. Students register for a problem solving session when they register for the course. Each problem solving session has up to fifty students. In these sessions, students learn the problem solving topics and practice the concepts individually or in small groups. Each problem solving session is one hour long. A second faculty member (different from the lecturer and lab coordinator) conducts the problem solving sessions. Problem-solving topics presented in a typical semester include:

1. Probability and Distributions
   Students are introduced to normal, uniform, and Poisson distributions. Students are expected to solve problems using these distributions including an understanding of basic statistical terms such as the mean, standard deviation, and mode.

2. Electrical Circuits

   Basic circuits are introduced and students are taught how to find the equivalent resistance of a circuit, voltage drops across resistances, and current through parallel branches using Ohm's Law.

3. Projectile Motion

   Students are taught how to calculate the acceleration, velocity, and position in both x- and y-directions of projectiles in motion ignoring friction and wind.

4. Taylor Series Approximations

   An introduction to expanding functions using Taylor's Theorem is presented with a focus on trigonometric functions such as sine, cosine, and tangent. Students are expected to perform estimations using Taylor expansions and to evaluate the estimates using Taylor's remainder function.

5. Linear Interpolation

   Given a set of data, students are taught how to linearly interpolate values for a function given inputs that do not have explicit outputs.

6. Relational Database Design and Data Mining

   An introduction to designing tables and queries are given including first, second, and third normal forms. Students are expected to design a basic database schema given a problem.

A problem-solving topic is introduced approximately every two weeks. During the first week a new topic is introduced with examples and a homework assignment. The second week includes a question-answer period along with hands-on practice.

In the Laboratory

In the lab students work in groups of two (or three if there is an odd number of students or a missing partner). Students spend two hours per week in the lab with between fifteen and twenty students per section. A graduate or undergraduate student is the dedicated lab instructor for the lab section. A third faculty member (in addition to the lecturer and problem-solving instructor) also holds the position of laboratory coordinator. This faculty member participates in the lab on a regular basis rotating the lab sections they are working with weekly. A student in a lab section would experience a lab environment with a dedicated lab instructor and the assistance of a faculty member every two or three weeks.

The labs used in this course give students practice in solving problems in engineering and computer science by developing computer programs. Students are required to submit a lab report for each lab. A single lab report is collected per group of students and is typically between one and three pages long. Students are encouraged to keep their lab reports concise and short. The lab reports are graded based on students' explanation of the problem and new

controls, properties, events, and Visual Basic syntax introduced in the lab. A typical semester of labs is listed below including a bullet point of new topics.

1. Using Visual Basic: Hello World
    - Forms, text boxes, labels, command buttons, and the click event.
    An introductory lab where students place their name in a text box and click a command button to see their name in a sentence welcoming them to Visual Basic.

2. Fat Estimation and Ideal Weight: How to Calculate Body Fat
    - Radio buttons, Frames, the Visible property, CInt, CDbl, and CStr, and a basic If statement.
    - How to calculate percent body fat and ideal weight for males and females [1].
    In this lab, students make a fat estimator that will calculate and display a users body fat and ideal weight based on specific body measurements. Students use a basic conditional statement and are introduced to making calculations in Visual Basic.

3. Using Random Numbers to Verify the Normal Distribution
    - A basic For-loop and the Rnd() function.
    - Mean and standard deviation.
    A simple program is developed to calculate the mean and standard deviation of random numbers. The user chooses how many random numbers will be used and the program computes the mean and standard deviation. The larger number of random numbers used, the closer the mean gets to 0.5. Since the data are not being stored anywhere, students must use the following form to compute the standard deviation of the data:

$$\sqrt{\frac{\sum_{i=1}^{n}(x_i)^2}{n} - \mu^2}$$

    where x is a random number, n is the number of samples, and μ is the mean. Students explain this in their lab report.

4. A Computerized Game of Craps
    - Randomize, Images, and variables with form scope.
    Students implement a game of Craps.

5. Circuit Analysis in Visual Basic
    - Lines, check boxes, the Slider, and Change event.
    - Analyzing a circuit using Ohm's Law.
    Students are required to derive current and voltage equations for two different circuits. The user selects which circuit they wish to analyze, then can dynamically change the resistance values in corresponding text boxes and the input voltage controlled by the slider.

6. Projectile Motion and Animation in Visual Basic
    - The Timer control, Timer event, and writing functions.
    - Finding *x*- and *y*-positions of a projectile in motion.
    In this program, the user selects a velocity using a slider to try and hit a randomly placed target. Students must write a function to move the projectile using the Timer's Timer event. Students must also adjust the equations they derive to compensate for the pixel coordinate system.

7. Graphing in Visual Basic using Taylor's Theorem
    - Microsoft Chart and loops.
    - Taylor's expansion for sine or cosine and factorial.
    Students write functions for computing the factorial of a number and for computing an approximation to sine or cosine using an *n*-order polynomial. Using the Microsoft Chart control, the approximation, actual values computed by the built-in trigonometric functions, and error between the two functions are graphed between 0 and 2π.

8. Finding Anagrams using a Data Histogram
    - One-dimensional arrays and functions, file input, string manipulation, and list boxes.
    - Anagrams and a data histogram.
    Students write functions to fill an array where each index represents the frequency that a letter appears in a sentence and to compare two such arrays to determine whether one phrase could be made from another (an Anagram). Possible Anagrams are read from a file. Anagrams are added to a list box.

9. Data Mining and Medical Applications using a Microsoft Access Database
    - Data control.
    Students are given a Microsoft Access database with a table filled with data from Hepatitis patients [2]. Four features from blood tests: *Billrubin, SGot, Alk Phosphate,* and *Albumin* are used. Students calculate the mean and standard deviation for each of the four features for patients who died and patients who lived. This information leads to an apparent separation of data allowing predictions of whether patients will live or die based on these tests.

10. Real-time Data Collection and Interfacing with Laboratory Equipment
   - Microsoft Comm control.
    Students write a program that collects data from an electronic thermometer [3] in real time. Students receive an ASCII string from the device, parse the string to extract the temperature, validate the temperature data as valid data, and graph the data on a Microsoft Chart.
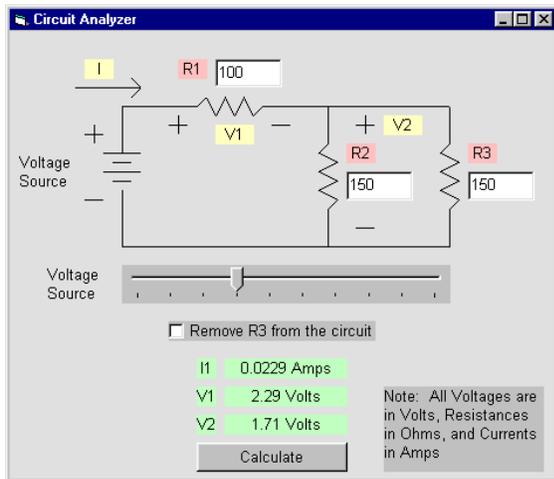
Labs 1, 2, 3, 4, 5, 7, 9, and 10 are one-week labs while labs 6 and 8 are two-week lab assignments. Labs begin the second week of class so that sufficient material can be presented to complete the first lab assignment. Figure 1 shows screen shots from selected labs.
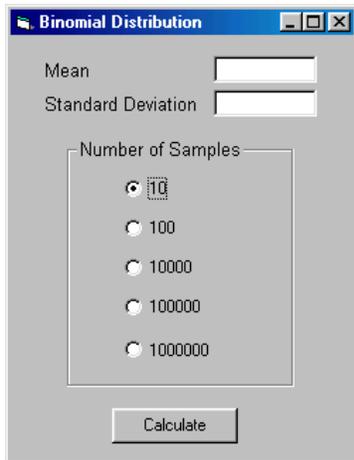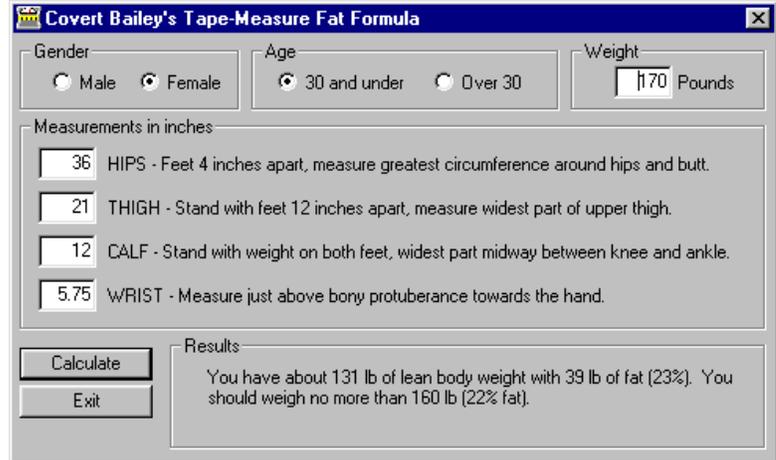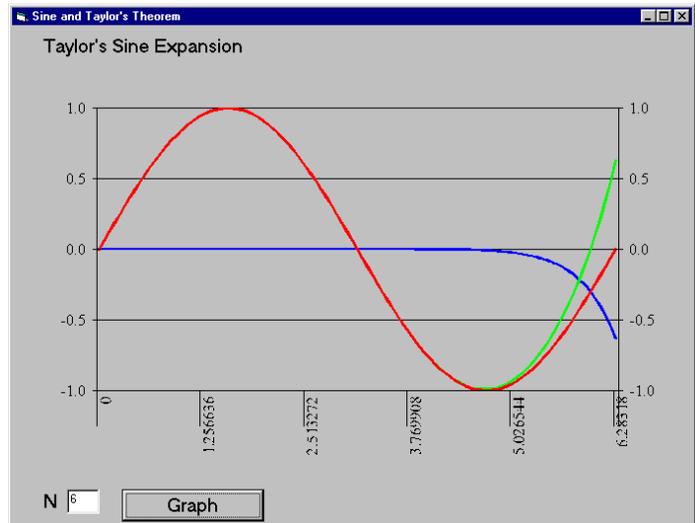
Lab 3   Lab 2

Lab 5   Lab 7

Figure 1: Screen shots from selected lab assignments

Educational Value

Computer Science and Computer Engineering majors who have had CSE 141 are much better prepared to tackle the rigors of the C++ language in CSE 230 (Object-Oriented Programming I). They are able to perform at a substantially higher level than students who entered the previous CS1 course (CSE 131 - Computing I) without having had CSE 141 as a prerequisite. With CSE 141 as a prerequisite to a Computing I course in an object-oriented language, students are prepared to construct algorithms to solve problems. This gives the instructor the opportunity to focus on the object-oriented aspects of C++ or Java without having to teach students basic constructs of programming and solving problems.

For all other Engineering majors who may never take another programming course, CSE 141 provides useful lifelong skills for rapidly developing applications to solve real-world engineering problems including database driven applications and controlling laboratory equipment with a visual interface that is appealing to users.

Conclusion

A new course, CSE 141, *Computer Problem Solving in Engineering and Computer Science*, was introduced at Oakland University in the fall of 2001. As the title of the course suggests the emphasis of the course is on problem solving using computers. Visual Basic was chosen for the programming language in this course because it is easy to learn and will be useful for all engineering students even if this is the only programming course they take. The text used in the course [4] emphasizes problem solving using Visual Basic.

The course has the flavor of a freshman introduction to engineering course taught at Oakland University over 30 years ago in which students solved interesting problems from all areas of engineering. The present course brings these ideas up to date by providing students with computer skills that will enable them to create interactive, windows-based programs that will be useful for them in solving engineering problems in the future.

References

1. Bailey, Covert, *The Ultimate Fit or Fat: Get in shape and stay in shape with America's best-loved and most effective fitness teacher*, Houghton Mifflin, 1999.
2. Cestnik, G., I. Konenenko, & I. Bratko, Assistant-86: A Knowledge-Elicitation Tool for Sophisticated Users. In I.Bratko & N.Lavrac (Eds.) Progress in Machine Learning, 31-45, Sigma Press, 1987.
3. Omega HH509R 2-Input Handheld Thermometer with RS-232 interface found at *www.omega.com*.
4. Alka Harriger, Susan Lisack, John Gotwals, and Kyle Lutes, *Introduction to Computer Programming with Visual Basic 6: A Problem-Solving Approach*, Que Education and Training, 1999.

Biography

Darrin M. Hanna, is Visiting Instructor of Engineering in the Department of Computer Science and Engineering at Oakland University. His primary area of research is pattern recognition and artificial intelligence and embedded systems.

Jerry E. Marsh, is Special Instructor of Engineering in the Department of Computer Science and Engineering at Oakland University. His primary areas of research are computer science education, object-oriented programming, and event-driven programming.

Richard E. Haskell is Professor of Engineering in the Department of Computer Science and Engineering at Oakland University. He is the author of 15 books and has taught numerous undergraduate and graduate courses including courses in microprocessors, embedded systems and digital design using VHDL.