Learning to program requires much hard work and dedication. Most students in introductory programming courses struggle to grasp programming concepts in general. While some students keep at it until the key concepts gel, many simply give up instead. Universities in the US, Canada, and elsewhere have reported withdrawal, failure, and D-grade rates approaching 50 percent in introductory computer programming courses.

To address these problems, we recently designed and successfully offered a revolutionary one-semester course that integrates the use of HTML, JavaScript, and Java. This approach departs markedly from using a single general-purpose programming language such as Java or C++.

COMMON PROBLEM

Today, most introductory programming courses are based on learning a general-purpose programming language such as Java. Yet many students with no previous computing background find it hard to grasp the concept of programming in general, and object-oriented programming in particular. Our experience shows that many students have no problems with the language itself, but the tools they are using—such as the development environment—make the learning process difficult. For example, the compiler often produces cryptic error messages.

Further, in introductory programming courses assignments usually consist of command-line input and output. Novice programming students struggle to interpret such cryptic error messages. This problem cannot be solved by teaching GUI programming in Java using, for example, the AWT/Swing packages, because programming in Swing requires a basic knowledge of Java syntax and its object-oriented programming mechanisms.

UNIQUE SOLUTION

The University of Guelph-Humber (www.guelphhumber.ca), an initiative between the University of Guelph and Humber College Institute of Technology and Advanced Learning, offers the Distributed Computing and Communications Systems Technology program. The two institutions combine to give students a fully recognized university honors degree in applied computing and a college diploma in communications systems technology. This four-year cooperative education program is the only one of its kind in Canada.

Students here can study the convergence of computing and telecommunications in a single program, which gives them the applied computing expertise they need to provide working solutions to real-life problems. The program emphasizes systematic approaches to the design and development of secure distributed applications, hands-on experiments in leading-edge wired and wireless technologies, and teamwork projects. In this lab-intensive program, students learn by doing.

To help the program succeed, an innovative curriculum encourages further learning and increases student retention. Our first computer programming course provides a step-by-step approach to programming that requires no previous experience with computers in general or programming in particular.

We designed the course to encourage students to get started with programming, even at a slow pace, rather than risk frustrating and eventually losing them. This course focuses on using computers to build applications for fun: static homepages, interactive homepages using JavaScript, and simple stand-alone Java applications. We accomplished this through click programming, a teaching methodology that integrates more than one programming language. By introducing HTML, JavaScript, and Java successively, each topic prepares the students...
for the next, harder topic. The course lasts 13 weeks, with a three-hour lecture each week plus a total of 10 three-hour labs. The course covers HTML for two weeks, JavaScript for four weeks, and Java for seven weeks.

The course’s first section introduces file management and basic Internet concepts, the World Wide Web, and HTML. The second section introduces the basic elements of programming in JavaScript, including input/output, decisions and control flow, instantiating objects, invoking methods on objects, defining functions, and handling events. The third section introduces basic Java concepts and the elements of programming in Java, including using primitive data types, instantiating and using objects, invoking methods, dealing with input and output but not file I/O, and designing classes.

We use pair programming (www.pairprogramming.com) for both labs and assignments. This approach lets students share knowledge and learn about tips and tricks that their classmates might be familiar with. In the pairing process, we avoided letting students pair themselves, fearing that friends and possibly strong programmers would pair together. To ensure a fair pairing process, we paired students randomly and rotated pairs for all labs and assignments.

OLD VERSUS NEW APPROACHES

Traditional introductory courses, which focus on teaching a general-purpose language, place several obstacles between students and successful learning—and increase the instructor’s workload as well.

General complexity

Several general-purpose programming languages, such as Pascal, Modula-2, C++, and Java, have been widely used in introductory programming courses. Teaching students how to program in any of these languages is difficult, especially if they have no previous computing background. For example, consider the following segment of Java code for the classical Hello World program:

```java
public class Hello {
    public static void main(String argv[]) {
        System.out.println("Hello World!");
    }
}
```

Even in this simple program, many things must be explained in the first lecture. For example, what is the meaning of public, static, void, main, String, and []? These unknowns could frustrate students immediately in the first week of the course.

In addition, user input and output must be explained in the first few weeks because students will need to write programs that depend on user input. Reading input from the user, however, is difficult. Consider the following segment of Java code that prompts the user to enter a number:

```java
System.out.print("Enter a number: ");
var number = parseInt(str);
```

When this JavaScript script runs in a browser, it will present the user with an accessible and attractive dialog box. Students certainly like using such powerful features because they immediately feel productive—an important benefit for first-year students that encourages them to pursue further learning. To use JavaScript, however, students must learn about HTML because JavaScript is embedded in HTML documents.

Axing the compiler

When a programmer writes in a conventional programming language such as Java, that code must be compiled and run. If any compilation errors occur, the programmer must edit the program, recompile, then run the program again. Making errors in a conventional programming language thus produces a fatal error that prevents the program from executing. Making errors in HTML is not fatal: The browser will make its best effort to render the page, but will probably not display it as intended. We did, however, encourage students to write valid HTML documents and not rely on the forgiveness of the browser. It is worth noting that students developed their HTML documents using a text editor of their choice—most used Windows Notepad and didn’t rely on an HTML editor.

Once students become comfortable with HTML, the course introduces them to some programming concepts using JavaScript. Students can embed their JavaScript code in HTML documents, a powerful approach that lets them learn programming at a slow pace without worrying about cryptic compiler-error messages.
Next, the course covers JavaScript. During this time, students learn about objects and method invocation. After this, they’re well prepared to move on to Java. Given that decisions and control-flow statements have already been covered in JavaScript, when the students move to Java they can start producing interesting Java programs almost immediately.

On the other hand, Java and JavaScript differ significantly with regard to their object models, execution environments, and typing strictness. These differences, especially typing, caused some confusion during the first week of our Java section. This confusion has never become a major problem, however, and students gain a good understanding of typed languages as well as implicit and explicit casting.

We strongly believe that teaching computer programming in the context of simple client-side Web applications provides a motivating framework for students and encourages them to excel. We urge computing educators to try this revolutionary programming approach.

Based on student retention rates, we believe our course has been successful. Overall, performance in programming assignments and exam results have far surpassed those in other introductory programming courses we have taught at universities with students of similar quality. In our first offering during the autumn of 2002, all 34 students passed the course. Student marks spanned a high of 96 percent and a low of 60 percent, averaging 73 percent. Students learned how to create simple and dynamic Web sites and acquired the fundamentals of programming.

Further, feedback collected from students suggests they enjoyed the experience. We believe this approach is applicable not only to novice computing professionals, but also to those who wish to gain more control over what their computer does for them. The course could also enhance the teaching of computing studies in schools. In an effort to increase enrollment in the Distributed Computing and Communications Systems Technology program, we plan to let students from other disciplines take the course.

Wusay H. Mahmoud is an assistant professor in the Department of Computing and Information Science, University of Guelph, and associate chair of the Distributed Computing and Communications Systems Technology program, University of Guelph-Humber. Contact him at qmahmoud@cis.uoguelph.ca.

Włodek Dobosiewicz is a professor and chair of the Department of Computing and Information Science, University of Guelph. Contact him at dobo@cis.uoguelph.ca.

David Swayne is a professor in the Department of Computing and Information Science, University of Guelph, and program coordinator of the Distributed Computing and Communications Systems Technology program, University of Guelph-Humber. Contact him at dswayne@cis.uoguelph.ca.

Editor: Neville Holmes, School of Computing, University of Tasmania, Locked Bag 1-359, Launceston 7250; neville.holmes@utas.edu.au

---

IEEE Computer Society members work together to define standards like IEEE 802, 1003, 1394, 1284, and many more.

Help shape future technologies • Join an IEEE Computer Society Standards Working Group at computer.org/standards/