Paper ID #32046

Using the free Coral language and simulator to simplify first-year programming courses

Prof. Frank Vahid, University of California, Riverside

Frank Vahid is a Professor of Computer Science and Engineering at the Univ. of California, Riverside. His research interests include CS/engineering education, and embedded systems. He is a co-founder of zyBooks.com.

Joe Michael Allen, University of California, Riverside

Joe Michael Allen is a Ph.D. student in Computer Science at the University of California, Riverside. His current research focuses on finding ways to improve CS education, specifically focusing on introductory programming courses known as CS1. Joe Michael is actively researching the impact of using a many small programs (MSP) teaching approach in CS1 courses. His other interests include educational games for building skills for college-level computer science and mathematics.

Dr. Alex Daniel Edgcomb, zyBooks

Alex Edgcomb is Sr. Software Engineer at zyBooks.com, a startup spun-off from UC Riverside that develops interactive, web-native learning materials for STEM courses. Alex is also a research specialist at UC Riverside, studying the efficacy of web-native content and digital education.

Prof. Roman Lysecky, University of Arizona

Roman Lysecky is a Professor of Electrical and Computer Engineering at the University of Arizona. He received his Ph.D. in Computer Science from the University of California, Riverside in 2005. His research focuses on embedded systems with emphasis on medical device security, automated threat detection and mitigation, runtime adaptable systems, performance and energy optimization, and non-intrusive observation methods. He is an author on more than 100 research publications in top journals and conferences. He received the Outstanding Ph.D. Dissertation Award from the European Design and Automation Association (EDAA) in 2006, a CAREER award from the National Science Foundation in 2009, and nine Best Paper Awards. He is an inventor on one US patent. He has authored eight textbooks on topics including C, C++, Java, Data Structures, VHDL, and Verilog, and he has contributed to several more. His recent textbooks with zyBooks utilize a web-native, active-learning approach that has shown measurable increases in student learning and course grades. He has also received multiple awards for Excellence at the Student Interface from the College of Engineering at the University of Arizona.

Using the Free Coral Language and Simulator to Simplify First-Year Programming Courses

Abstract

Many engineering majors require first-year students to learn programming. Unfortunately, commercial languages like Python, C, C++, and Java were designed for professionals, not learners, and thus have nuances that can cause students to struggle. Such struggle can lead to frustration, low grades, and potentially to dropping their programming classes or even switching majors. The Coral language was created in 2017 to address this issue. Coral is ultra-simple, looking almost like pseudocode, with fewer than 10 instruction types. Coral has a free web-based educational simulator, which auto-derives a graphical flowchart, and which executes the code and flowchart visually while showing variable updates in memory. Unlike other educational programming environments like Alice, Scratch, or Snap, Coral was designed for college students, with an emphasis on leading smoothly into a commercial language. Though Coral is now used by many thousands of students in CS0 classes at dozens of universities, in Fall 2019 our university experimented with introducing Coral in its CS1 class, where one 80-student section was taught programming in Coral for the first 5 weeks, then C++ for the second 5 weeks. Those Coral-to-C++ students did equally well on the identical C++ final exam compared to the students in other class sections who learned C++ the entire term, and their code style was better. Coral-to-C++ students' evaluations were also very positive, and teachers reported an exceptionally smooth class startup using Coral. The C++ class sections were already highly optimized with strong performance and excellent student evaluations. These Coral-to-C++ results suggest that Coral can be used to enable a simpler and smoother start to a freshmen programming class, while still achieving the desired learning of a commercial language. And, as the Coral approach is improved, one might begin to see Coral-to-C++ students outperforming C++-only students as well. The Coral simulator and tutorial are available for free online [1].

Introduction

Coral [1] is an ultra-simple text and flowchart language designed to introduce college or high-school students to programming. In contrast to many educational programming languages that use blocks, like Scratch [2] or Snap [3], Coral is specifically intended to lead students into commercial languages like Java, Python, C++, or C.

Coral was introduced in 2017 as a joint project by the University of California at Riverside, the University of Arizona, and zyBooks [4]. A key feature of their offering is a free educational simulator at CoralLanguage.org, along with a tutorial. The simulator auto-derives the flowchart from the text, laying out the flowchart to closely match the code. The simulator allows step-by-step execution, highlighting each text statement or flowchart node. The simulator shows variables in memory, inputs being consumed, and output being generated on a simulated screen. Coral has been shown to improve a CS0 class at the University of Arkansas that previously had students drawing flowcharts on paper [5]. For further information on Coral, we refer the reader to Coral's website [1] or to read the introductory paper by Coral's authors [6]. Figure 1 shows a sample program in Coral's ultra-simple text syntax, and an equivalent flowchart syntax.

```
integer x
integer y
integer max

x = Get next input
y = Get next input

if x > y
   max = x
else
   max = y

Put max to output
```

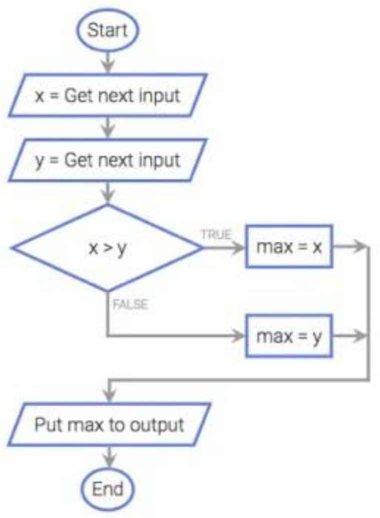

Figure 1. (Left) Sample Coral program using Coral's ultra-simple text syntax. (Right) Same sample program using Coral's flowchart syntax (auto-derived from the text by the simulator).

In this work, we sought to determine the feasibility of using Coral in a traditional CS1 class that teaches C++. Our university is a major public university with 25,000 students, typically ranked in the top 50-75 among all U.S. universities. Our CS1 teaches C++ to 300-500 students per 10-week quarter, broken into sections of about 80 students. All sections are usually kept in sync, using the same syllabus, textbook, assignments, exams, etc., even though different sections might have a different instructor. In Fall 2019, for one 80-student section (the "Coral-to-C++" section), we introduced Coral for the first 5 weeks of the class up to the midterm, and then switched to C++ for the last 5 weeks. Those students did many of the same C++ assignments in the latter half of the course as the C++-only sections and took the same final exam as the other sections. Our engineering college uses "learning communities" such that all computing-major freshmen are scheduled into specific sections; to avoid bias, the Coral-to-C++ section was not one of those sections, but rather was a section almost entirely with non-computing majors.

Starting with Coral has big advantages for instructors and students. For students, nearly all focus is on learning programming concepts and solving problems using programming logic; the Coral language's simplicity and the informative messages from the Coral simulator eliminate nearly all early tool and syntax issues. For instructors, not only is the initial workload reduced due to fewer questions/worries, but instructors can use the simulator to visually demonstrate step-by-step code execution, flowchart representations of branches and loops, variables being updated in memory, arrays being created in memory, and function call creating a local execution situation. But, questions remain as to whether starting in Coral and switching to C++ yields to equally proficient learning of C++, and whether that approach will be well-received by students, who might be frustrated not learning a "real" language or stressed by the conversion from one language to another mid-term. This paper presents performance data and survey data comparing the Coral-to-C++ section to the C++-only section to address those questions.

Performance: Grades

Our first analysis was to compare student performance with respect to attained grades between the C++-only section and the Coral-to-C++ section. We collected the gradebooks for each section at the end of the quarter and averaged student grades across the grading categories of the class. The total grade was broken into categories: Midterm exam, Final exam, Participation activities (PAs, meaning readings and online learning questions), Challenge activities (CAs, meaning homework problems), Lab activities (LAs, meaning programming assignments), and class participation. As class participation grade accounts for only a small portion of the overall grade and everyone achieves nearly 100%, we excluded participation from the analysis.

Figure 2 shows average grades (y-axis) for each category (x-axis), with the left bars for C++-only sections and the right bars for the Coral-to-C++ section. The midterm and final exams are also shown broken down into their multiple choice (MC) and coding parts (each exam's time and points was half MC and half coding). Students that did not take the midterm exam or final exam were excluded, as they likely dropped the class. Significance values from a 2-tailed t-test are shown below each category.

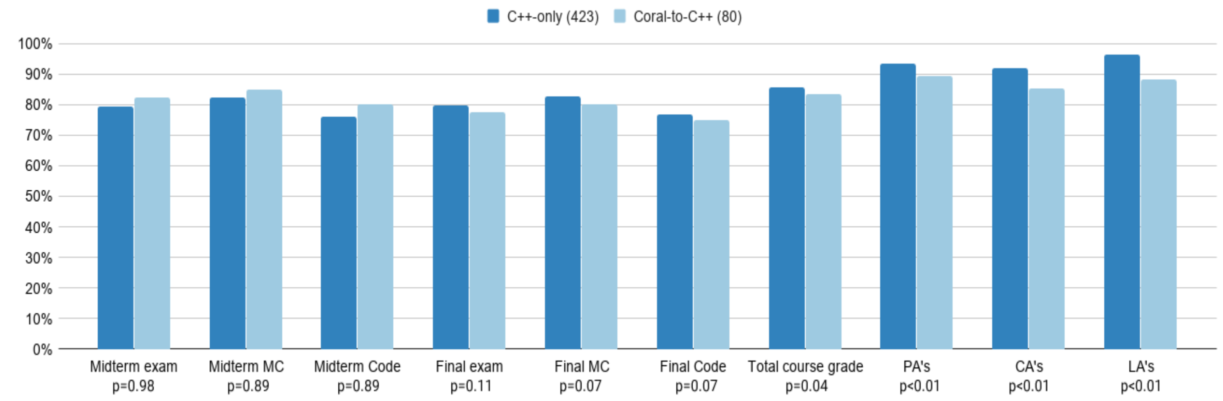

Figure 2. Grade performance analysis comparing the C++-only section with the Coral-to-C++ section.

Figure 2 summarizes the results of the grade performance analysis. Looking specifically at the identical C++ final exam that all sections took, the difference is not statistically significant (p = 0.11, which is not < 0.05). The small difference is even less relevant given that the Coral-to-C++section was mostly non-computing majors, and thus usually slightly underperforms the sections that have most of the computing majors.

We do see in Figure 2 that the Coral-to-C++ section scored significantly lower on the PAs, CAs, and LAs. We realized the cause to be excessive workload -- we packed too much C++ into the second half of the class, much of it redundant with the Coral, so many students decided to not do all the assigned work in the second half of the term. Because PAs, CAs, and LAs contribute to 30% of the course grade, the overall course grade was lower too. But the learning outcomes as measured by the final exam suggest that C++ itself was learned just as well. In our most recent offering, we have corrected for the excessive workload in the second half of the term.

Stress survey

For several years, we have given CS1 students weekly surveys in lecture and lab. In week 8, we give what we internally call a "stress survey" that aims to learn how students are feeling about the class – stressed, confident, happy, worried, etc. The stress survey has 18 agreement statements and a 6-point Likert Scale: Strongly agree (5), Agree (4), Slightly agree (3), Slightly disagree (2), Disagree (1), and Strongly disagree (0). To reduce bias, some questions are phrased so that agreement is more favorable ("I enjoy the class") while others so that disagreement is more favorable ("I am anxious about the final"). Table 1 shows the questions and average response value for each question. For readability, Table 1 is organized with all agreement-favorable questions first, and disagreement-favorable questions second, but importantly, students received the questions in a randomized order. More favorable responses for each question have been shaded grey. A '*' has been added to the p-value column if the response is significant ($p < 0.05$).

Table 1. Results from the "Stress survey" given to students in week 8.

	C++-only	**Coral-to-C++**	**p-value**
I enjoy the class.	3.67	4.06	0.02*
This class is an appropriate amount of work per week for the number of units.	3.49	3.28	0.24
I was prepared for the midterm exam.	3.28	3.49	0.31
I feel prepared for the final exam.	2.53	2.30	0.26
The weekly zyLab programming assignments were enjoyable.	3.10	3.02	0.67
The weekly zyLab programming assignments contributed to my success in the course.	3.68	3.82	0.36
I learned a lot from the weekly zyLab programming assignments.	3.77	3.91	0.32
I frequently collaborated with others on the weekly programming assignments.	2.40	1.54	<0.01*
I feel confident in my ability to write a small (< 50 line) useful program.	3.26	3.17	0.71
I am often anxious about the class.	2.47	2.63	0.46
I spend a lot of time in the class figuring out system issues rather than learning programming.	2.02	1.74	0.16
The number of tools and websites for this class are somewhat overwhelming.	2.00	1.89	0.58
I have missed a deadline because I thought it was another time.	1.98	2.13	0.57
I have looked for class info but couldn't find it.	1.77	1.31	0.01*

I felt anxious before the midterm exam.	3.39	3.45	0.79
I feel anxious about the final exam.	3.81	4.06	0.17
The weekly zyLab programming assignments were stressful.	2.60	3.12	<0.01*
The weekly zyLab programming assignments were frustrating.	2.73	3.26	<0.01*

Table 1 shows the responses are similar for both sections; most differences are not statistically significant. Questions with $p < 0.05$ show that the Coral-to-C++ section enjoyed the class more, and found all class information easier, while the C++-only section collaborated more and found the LA's to be less stressful and frustrating. As mentioned in the previous section, we believe the Coral-to-C++ class found the LA's more frustrating due to excessive workload in the second half of the term.

Student survey feedback on the Coral/C++ experiment

In addition to our weekly surveys including the stress survey highlighted above, we gave the Coral-to-C++ students a special survey at the end of the term. Questions were agreement statements with a 5-point Likert Scale: Strongly agree (4), Agree (3), Neutral (2), Disagree (1), and Strongly disagree (0). There were also several free response questions. 64 of the 80 total students responded.

The results suggest that students enjoyed learning Coral (3.1) and saw the value of learning programming concepts like loops or functions in Coral before C++ (2.86). Students did not struggle with Coral syntax (1.31), they found the Coral simulator helpful (3.23), and they thought learning Coral first made C++ easier to learn (2.75). However, students were neutral on preferring to learn C++ over Coral first (1.97) and whether or not learning Coral first caused them to struggle more with C++ syntax (1.54). Overall, the responses indicate that students were generally happy, and the Coral-to-C++ approach did not bother them (even though we gave them too much work in the second half of the term).

One free response question was: "Please comment on the approach of learning Coral before C++ (pros, cons, experiences, etc)." The comments were quite positive, such as those below.

- “Learning the concepts we learned in Coral in C++ was like extra review. (was easier to understand and make a connection with).”
- “It was the best! I actually understood functions better the 2nd time around. Even though some coding was different such as having to use "cout" instead of "put to output" it was still easy!”
- “Learning Coral first was AMAZING. It made learning concepts so much easier since the language and form was intuitive! Then C++ concepts were easy, and we could focus on mastering language specifics.”

The few negative comments were not really negative. They mostly came from students who had prior programming experience and would have preferred to jump into C++, but who seemed to understand and accept the role of Coral. Such comments included those below.

- “Coral wasn't my first programming language but I do think that its psuedocode style syntax is more helpful for introducing fundamental concepts as it gives more of a general introduction as opposed to the "here's how to do this in this specific language" style of teaching.”
- “Since I already had experience programming before it didn't make too much of a difference to learn Coral before C++ but I could see why it would be easier for people who had no coding experience to learn Coral before C++.”

There were also a few negative comments about feeling rushed and overwhelmed in the final weeks of learning C++, which we attribute to our own mistake of assigning too much work. We have since corrected that issue in the most recent offering.

Conclusion

Starting a CS1 class by teaching Coral and then switching to a commercial language like C++ has numerous advantages: For students, Coral eliminates nearly all tool and syntax issues in the first half of a class, while for instructors, demand for help is greatly reduced plus instructors can use Coral's powerful educational simulator to help students understand each new construct and how programs execute. In this work, we showed that a Coral-to-C++ approach does not hamper learning of C++; students performed nearly identically on a C++ final exam (multiple choice and coding) as students who took CS1 entirely in C++. Furthermore, we showed that, rather than being frustrated not learning a commercial language initially or stressed due to switching languages mid-term, instead students were generally quite happy with the Coral-to-C++ approach.

References

[1] Coral coralLanguage.org, 2017. (Accessed May 2020).

[2] Scratch https://scratch.mit.edu/. (Accessed May 2020).

[3] Snap https://snap.berkeley.edu/. (Accessed May 2020).

[4] zyBooks https://www.zybooks.com/catalog/zylabs-programming/. (Accessed: May 2020).

[5] A. Edgcomb, D. McKinney, F. Vahid, R. Lysecky, "Improving Pass Rates by Switching from a Passive to an Active Learning Textbook in CS0," ASEE Annual Conference, 2020.

[6] A. Edgcomb, F. Vahid, and R. Lysecky, "Coral: An Ultra-Simple Language For Learning to Program," Proceedings of ASEE Annual Conference, 2019.