Does Story Programming Prepare for Coding?

Jennifer Parham-Mocello
School of EECS
Oregon State University, USA
parhammj@oregonstate.edu

Martin Erwig
School of EECS
Oregon State University, USA
erwig@oregonstate.edu

ABSTRACT

In this research study, we investigate the impact of using the Story Programming approach to teach CS concepts on student performance in a subsequent C++ class. In particular, we compare how students receiving little or no coding to learn and apply these concepts perform in comparison to students who learn these concepts only in the context of coding. While past research has shown that exposure to programming is not a predictor of success in such courses, these studies are based on a 15-week versus 10-week course and do not control for the CS concepts and programming to which the students have been exposed. Consequently, we hypothesize that students from the Story Programming approach will perform worse in the following C++ class. Surprisingly, we find that this is not true: Students from the Story Programming approach with little to no coding do not significantly differ from their peers receiving a traditional code-focused approach.

CCS CONCEPTS

• Social and professional topics ~ Computational thinking
• Social and professional topics ~ Computer science education
• Applied computing ~ Interactive learning environments

KEYWORDS

Story Telling; Computer Science Pedagogy; Introduction to Computer Science

ACM Reference format:

Jennifer Parham-Mocello and Martin Erwig. 2020. Does Story Programming Prepare for Coding? *In the 51st ACM Technical Symposium on Computer Science Education (SIGCSE'20). ACM, New York, NY, USA, 7 pages.* https://doi.org/10.1145/3328778.3366861.

1 INTRODUCTION

Inspired by our previous research using stories to explain computation before coding [16], we want to know how students receiving this treatment with little or no coding to learn and apply CS concepts perform in a subsequent C++ class, especially in comparison to students who learn these concepts only in the context of coding. Even though the Story Programming approach is a viable alternative to a code-focused approach for CS 0, we hypothesize that students from the Story Programming approach will perform worse in the following C++ class:

> *"Students in the new Story Programming approach will perform worse in CS 1, since they lack a half (or even a full) term of coding to learn and apply CS 1 concepts."*

While past research shows that exposure to programming is not a predictor of success in such courses, these studies are based on a 15-week versus 10-week course and do not control for the CS concepts and programming to which the students have been exposed. In this paper, we compare the CS 1 performance of students who receive no coding or some coding using the Story Programming approach in CS 0 with students who receive a full 10-week term of coding. At the host university, CS 1 is a very large, challenging C++ class with 5-6 assignments and a heavy focus on pointers and memory management in the second half of the 10-week term. Over the years, the instructor, who teaches both CS 0 and CS 1, added more and more coding to CS 0 to "better" prepare students for CS 1, even though CS 0 is not a prerequisite for CS 1. This research study investigates whether students who receive more coding in CS 0 really are better prepared for CS 1 and whether the students in the Story Programming approach with little or no coding are at a disadvantage.

2 BACKGROUND

2.1 Unplugged Curriculum

Many approaches to introducing computer science to students are predicated on programming, that is, they require an understanding of how to code an algorithm in a programming language. This approach is used in efforts such as *code.org* that promote coding and computer science to younger children, but many studies have shown that comfort level and willingness to learn programming are the strongest predictors of success in programming classes [23, 18, 19]. Since computer science is not synonymous with programming, there is no inherent necessity to tie the orientation to computer science to coding activities, and students who are not sure whether they want to study computer science but are curious about the subject should not be excluded because they are reluctant to the idea of having to become a programmer as a prerequisite to understanding computer science.

This is why efforts to explain computer science without a computer, such as *csunplugged.org* [2], have gained popularity,

* This work is partially supported by the National Science Foundation under the grant CCF-1717300.

SIGCSE '20, March 11–14, 2020, Portland, OR, USA © 2020 Association for Computing Machinery. ACM ISBN 978-1-4503-6793-6/20/03…$15.00
https://doi.org/10.1145/3328778.3366861

especially among the K-12 community [1, 4, 8, 17, 21], and studies show that this approach broadens participation [3]. For these reasons, the researchers in this study believe that students and CS programs at institutes of higher education can benefit from non-coding alternatives.

2.2 Promoting Computational Thinking

Most alternatives for teaching introductory CS courses at the university focus on changing the curriculum or adding pathways to improve success and retention [9, 14, 11] or make topics covered more relevant and broader [13]. Some universities create interest-based classes allowing students to choose a class section based on what they like, such as game development, robotics, music, and mobile applications [9, 24], while others focus on adding computational thinking to their curricula with and without the use of a computer [7, 10, 14, 22]. These studies show that adding computational thinking to the curricula help students think about different ways to attack problems, making them more effective problem solvers, but these studies do not show whether the removal of coding or increased computational thinking activities helps or hinders students' success is subsequent coding courses.

2.3 Factors of Success in Programming Courses

There are many research studies on factors that predict student success in CS 1 and CS 2 courses, especially regarding prior programming experience [20, 18, 19, 23] and grade received in prior classes [5, 11, 12]. Wilson and Rountree find that prior programming experience is not a predictor of student success or failure in a 15-week intro to programming class, and instead, comfort level and expectations seem to play a larger role. Whereas Smith's research shows that an aptitude test measuring the retention of prior programming experience, rather than a self-reported survey, is correlated to the overall success in a 15-week intro programming class, Danielsiek, Kirkpatrick, and Lambert all show that the grade received in the prior CS0 or CS1 course is the strongest predictor of success in the subsequent class. These research studies suggest that intro CS courses focused on programming are not well suited for students who are not comfortable in a programming class or who do not really want to learn programming, and mere exposure to programming before these classes is not a strong predictor because self-reported experience does not always account for the amount, quality, and retention of experience. We extend these studies by comparing students' performance in a subsequent challenging, 10-week C++ coding class, CS1, by controlling for their experiences to specific CS concepts and amount of coding used to learn and apply these concepts prior to CS 1.

2.4 The Story Programming Approach

Traditionally, the students in CS 0 at the host university learn and apply CS 1 concepts, such as algorithms, representation, control flow (both conditional statements and looping), functions, and lists, in the context of Python coding. The students write small Python programs as early as week two in a ten-week term. The lectures focus on teaching basic Python, and students learn basic problem-solving skills by creating documents with designs of solutions to computer science problems and plans for testing before writing their code.

As we sought to understand whether students who receive more coding in CS 0 are at an advantage in CS 1, we created an alternative pedagogical approach to teaching these concepts outside the context of coding using stories to explain computation based on *Once Upon an Algorithm* [6], which we call Story Programming. For instance, students learn about path-finding algorithms using the story of Hansel and Gretel, and groups of students write and act out these path-finding algorithms with pebbles during lab.

Students read relevant chapters of *Once Upon an Algorithm* before class with a weekly online quiz. Chapters are generally covered in a sequential fashion with some occasional skipping to group like concepts together. The use of the book in the Story Programming approach changes the emphasis placed on some concepts covered in early CS classes, allowing for greater breadth and depth of concepts. For example, the Story Programming approach covers the concept of data structures and decidability more deeply than the traditional approach, which only gives a cursory view of data structures through lists and only mentions the idea that not all problems are solvable. The Story Programming approach can be used to delay or remove coding, but it is important that the CS concepts be explained, learned, and applied in the context of stories before simulating the CS concepts from the stories using code.

In the remainder of this paper, we report on a study that investigates whether students from the Story Programming approach with little to no coding do worse than other students in a traditional code-focused approach. We use students' course, assignment, and exam grades in CS 1 to measure performance, and surprisingly, students from the Story Programming approach do not significantly differ from their peers.

3 RESEARCH METHOD

This research study was conducted using five different sections of CS 0 from fall 2017 and fall 2018. The same instructor taught all sections, and each section had approximately 100 students. Each section had two 50-minute lectures, one two-hour lab, and one assignment each week. During lecture, all students in each section engaged in group exercises that did not involve a computer. The sections varied in approach, amount of coding, and whether students self-selected into the approach.

In fall 2017, we divided CS 0 into three sections to offer the new Story Programming approach. We taught one section using the traditional Python programming approach, and in the other two sections, we used the Story Programming approach and delayed coding for half a term. When coding and a language was introduced, one section used Python and the other used Haskell to evaluate which language offers better affordances in the subsequent CS courses. In this year, the students were unaware of the differences between sections to provide an unbiased view,

but due to many unhappy students, we decided to allow students to self-select into the approach in the following year.

In fall 2018, we taught another section using the Story Programming approach without any coding to see if students do any worse than those from the previous year, and we continued to teach the traditional Python section. This time, the course catalog reflected the different themes for the sections, and advisors gave students information about the sections during registration. We acknowledge the threat to validity this causes, but it is important to research the impacts the Story Programming approach has on students beyond those who only self-select into the approach.

3.1 Research Questions

With 2 years of data, we look at how students from different sections perform in the subsequent C++ coding course to determine if the Story Programming approaches put students at a disadvantage. To investigate this issue, we formulated the following research questions:

RQ1: Do CS 1 course grades differ based on the CS 0 section?

In particular, we ask

- Do the CS 1 mean course grades differ depending on the prior CS 0 section?
- Do students' CS 1 course grade gains or losses depend on the taken CS 0 section?
- Does the percentage of students who do not pass CS 1 with a C or better differ depending on the prior CS 0 section?
- Does the performance in a CS 0 section determine how likely a student is to pass CS 1?
- Do CS 1 passing grades of well-performing CS 0 students depend on the prior CS 0 section?

RQ2: Do CS 1 assignment and exam grades differ based on the CS 0 section?

Since RQ1 only gives a high-level view of the impact different sections have on the overall performance in CS 1, we ask RQ2 to determine the impact each CS 0 section has on applying specific CS 1 concepts covered in individual assignments and exams. We also keep RQ1, since it doesn't require student consent, and we therefore have access to a larger number of data points.

3.2 Data Collection

With IRB permission, student-level grade information for all the students over the last five years of CS 0 and CS 1 were collected from the registrar with unique, random ids to match students across years and classes, and the participant-level assignment and exam scores for consenting students in CS 0 and CS 1 were collected from the instructor.

4 RESULTS AND DISCUSSION

The course-level student success results are out of 355 students, and the student-level assignment and exam grades are out of 103 consenting participants. Since the data are not normally distributed, the non-parametric Kruskal-Wallis test is used to reject hypotheses with 95% confidence, α=.05.

RQ1: Do CS 1 course grades differ based on the CS 0 section?

In order to answer this question, we need to know which students go on to take CS 1 after the orientation course (see Table 1). Even though only 65-75% of the students from a section take CS 1 after CS 0, we find that 7-13% of students in a section took CS 1 prior to CS 0, which leaves approximately 15-20% who do not take CS 1 in the future (see Table 1). Most students who take CS 1 after CS 0 do so in the following winter quarter, and there is no statistical difference between student course grades in winter 2018 CS 1 and winter 2019 CS 1 (chi-squared = 0.80823, df = 1, p-value = 0.3686).

Section (Enrollment)	*Took CS 1 before CS 0*	*Took CS 1 after CS 0*	*Total with Grades in CS 0 and CS 1*
Story Python **(105)**	8.6% **(9)**	70.5% **(74)**	74-0-5=**69**
Story Haskell **(102)**	9.8% **(10)**	74.5% **(76)**	76-2-10=**64**
Traditional 17 **(111)**	6.3% **(7)**	64.9% **(72)**	72-1-5=**66**
Story No Code **(89)**	7.9% **(7)**	71.9% **(64)**	64-0-7=**57**
Traditional 18 **(102)**	12.7% **(13)**	67.6% **(69)**	69-3-1=**65**

Table 1: Total number of students enrolled in each section, and the percentage of those who take CS1 before and after CS0, with the number remaining who have grades in both.

Some of the students who took CS 1 after CS 0 withdrew from CS 0, which were only 1-3 students in the 3 sections (Story Haskell and both Traditional). We removed these students from the sections, since we are investigating correlations between the performance in CS 0 with performance in CS 1. As a side note, 3 out of the 6 who withdrew from CS 0 did not pass CS 1.

1.4-13.2% of the students (1-10 students) from each section withdrew or took the CS 1 class for satisfactory/unsatisfactory (S/U) credit only, with the Story Haskell section having the most and the Traditional 18 having the least. We also removed these students because we cannot report on how well they performed in CS 1. The passing grade to move on to CS 2 is a C or above, and an S grade can be given for a D- or above.

From the remaining students, there were 28 students from the international bridge program (INTO) in the Story Haskell section, which was unique to this section. The other sections in fall 2017 had only three or none. Since the removal of this demographic significantly changes the percentages in this section, we report statistics with and without the INTO students in the fall 2017 Story Haskell section. In fall 2018, the INTO students were evenly distributed across the sections.

Do the CS 1 mean course grades differ depending on the prior CS 0 section?

After we removed students without grades in both courses from the study, we use non-parametric statistics to determine if the mean course grades for students from different sections differ. The mean course grades did not differ by section with INTO (chi-squared = 8.907, df = 4, p-value = 0.0635) or without INTO (chi-squared = 9.416, df = 4, p-value = 0.052) (see Figure 1). This means that students from the Story Programming approach perform as well as those from the traditional approach.

Moreover, if the INTO students are removed, then the Story Programming approaches from the fall 2017 slightly outperform the traditional approach that term.

Do students' CS 1 course grade gains or losses depend on the taken CS 0 section?

While there is no difference in mean course grades, there is a difference in the course grade loss among students from different sections with INTO (chi-squared = 27.871, df = 4, p-value = 1.325e-05) and without INTO (Kruskal-Wallis chi-squared = 23.621, df = 4, p-value = 9.513e-05). Even though most students receive a lower grade in CS 1, the students from the traditional section from fall 2018 do not see a decrease in grade as much as the other sections (see Figure 2). This does not necessarily mean that the traditional approach is better. The traditional section from the fall 2017 does not perform better than the Story Programming sections, and the Story Haskell section without INTO students is not significantly different from the traditional section in fall 2018.

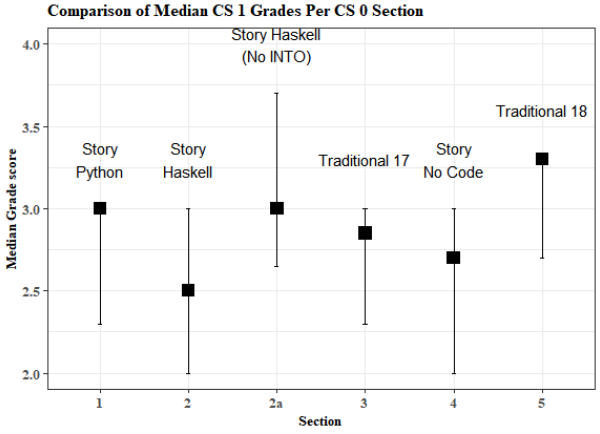

Figure 1: CS 1 Median course grades with upper and lower percentiles for students from different CS 0 sections.

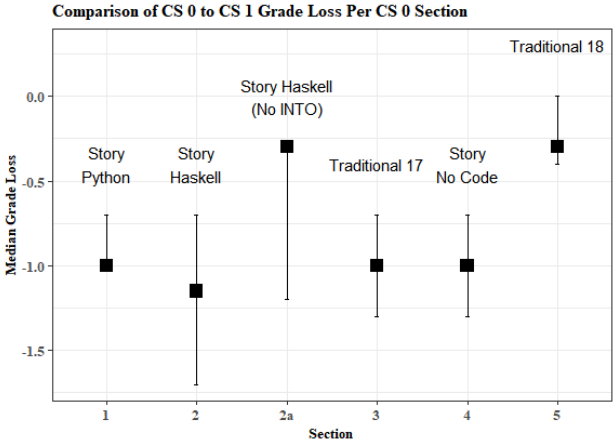

Figure 2: Median grade loss from CS 0 to CS 1 by section.

Does the percentage of students who do not pass CS 1 with a C or better differ depending on the prior CS 0 section?

Anywhere between 20-35% of students do not pass CS 1, and the traditional approaches from both years have the greatest number of students who pass CS 1 (see Table 2), which could be because the faculty member who developed the CS 1 curriculum also developed the traditional CS 0 curriculum. If the 28 INTO students are removed from the Story Haskell section, then both Story Programming sections delaying coding by a half term have about the same percentage of students as the traditional sections who pass CS 1. However, when the INTO students are removed from the Story Haskell section, the fall 2018 Story Programming section with no code has the highest percentage of students not passing CS 1 with a C or better. This might suggest that students who receive some coding to learn and apply CS concepts in CS 0 are more likely to pass the class but are not necessarily that much more likely to get a higher grade, as we see in Figures 1 and 2.

Does the performance in a CS 0 section determine how likely a student is to pass CS 1?

Since studies show that students who get below a B (3.0) or B- (2.7) in the introductory CS 0 or CS 1 courses are more likely not to pass the CS 1 or CS 2 class [5, 11, 12], we want to find out if this is true for our dataset. Out of 5 to 20% of the students in CS 0 who do not receive a B or better, we also find that most of these students are not likely to pass CS 1 (see Table 2).

All students in the fall 2017 Story Programming sections who did not have a B in CS 0, which is only 5.6-10.9% of the students, did not pass CS 1, and out of the 12% of the students who did not get a B or better in the traditional Python section in fall 2017, 62.5% did not pass CS 1 (see Table 2). In the fall 2018 CS 0 sections, the number of students with lower than a B was a little higher than in the fall 2017 classes, but similarly, 69-89% of those students did not pass CS 1 with a C or higher.

Our data show that those who do not receive a B or better in CS 0 have less than a 40% chance of passing CS 1, and this chance is even smaller for those who do not receive a B or better in the Story Programming sections, where coding is delayed or omitted. However, most of the students who do not pass CS 1 had a B or above in CS0.

Section (Total with Grades in CS 0 and CS 1)	***Do Not Pass CS 1 with ≥ C***	***Do Not Pass CS 1 and had < B in CS 0***	***Total Well-Performing Students***
Story Python **(69)**	23.2% **(16)**	6/6=**100%**	69-6=**63**
Story Haskell **(64)**	35.9% **(23)**	7/7=**100%**	64-7=**57**
Story Haskell (No INTO) **(36)**	25.0% **(9)**	2/2=**100%**	36-2=**34**
Traditional 17 **(66)**	22.7% **(15)**	5/8=**62.5%**	66-8=**58**
Story No Code **(57)**	29.8% **(17)**	8/9=**88.9%**	57-9=**48**
Traditional 18 **(65)**	20.0% **(13)**	9/13=**69.2%**	65-13=**52**

Table 2: Percentage of students from different CS 0 sections not passing CS 1 with a C or better, as well as those who do not pass CS 1 and had below a B in CS 0.

Do CS 1 passing grades of well-performing CS 0 students depend on the prior CS 0 section?

Since we know that the lower-performing students (those with lower than a B) in CS 0 are not likely to pass CS 1, we removed those students most likely not to pass CS1 from the data to find what the CS 1 grades are of well-performing CS 0 students (those with a B or better) (see Table 2).

Among the well-performing students with a B or above in CS 0, 72-92% (79-92% without INTO) of those students pass CS 1 with a C or above (see Figure 3). While the traditional section from the fall 2018 has the most students pass CS 1, there was not a significant difference among sections. Even though the Story Programming approach without any coding had the fewest number of students making above a B, the traditional section from fall 2017 had about the same number of students who did not make a B or better and even less students than the non-coding section receiving ≥ A- in CS 1 (see Figure 3). The traditional section from 2018 and the Story Programming approaches from the fall 2017 have about the same number of students with a B or above, and the traditional section from the fall 2018 term and the Story Programming Haskell section without the INTO students have the most students getting ≥ A-.

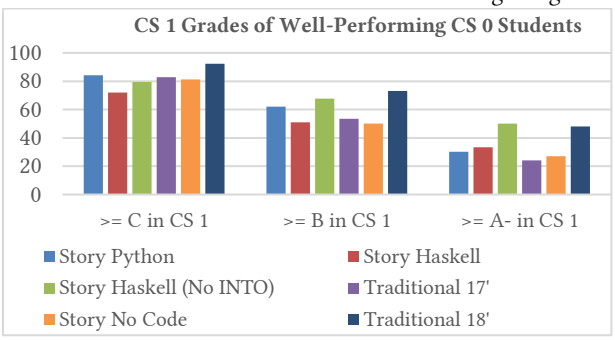

Figure 3: Percentage of students with a B or above in CS 0 and a C or above, B or above, or an A- or above in CS 1.

RQ2: Do CS 1 assignment and exam grades differ based on the CS 0 section?

To answer this question, we can only use students who consented in CS 0 and CS 1 to allow us to conduct research on their CS 0 and CS 1 assignments and exams. Approximately 20 students from each section consented in both CS 0 and CS 1, which we understand is not a large sample size. Most median course grades and upper/lower percentiles were very similar to the entire class, except that the Story Programming Haskell consenting students were more similar when INTO was removed (see Figure 4). Even though a Kruskal-Wallis test yielded a p-value < .05 for mean course grades of all students versus consenting students in each section, a multiple comparisons test with adjusted p-values did not yield any significant differences between the sections.

The consenting participants' mean CS 1 course grade, assignments, and exams did not differ by CS 0 section (see Table 3). This means that all Story Programming approaches (with and without coding) do as well as the traditional coding approaches. While not significant, we do notice that the students in the Story Programming section without code tend to do a little worse on the assignments and exams in CS 1, and the Story Programming approaches from fall 2017 that delayed coding by half a term always perform as well as or slightly better than the traditional approach on the exams (see Figure 5), but the traditional section from fall 2018 slightly outperforms students in all programming assignments.

As with any educational research study, there are threats to the validity of the results. Even though the course learning objectives and basic structure of CS 1 is the same in both years, the instructors, assignments, exams, and other course material differ, which could contribute to one year being more or less difficult than the other year, but the cohort of students could also contribute to this difference. In addition, we recognize that this study only includes consenting participants for the assignment and exam scores, but we minimize this threat by comparing the mean performance of non-consenting students with those consenting in CS 1 to make sure these values are representative of the whole class.

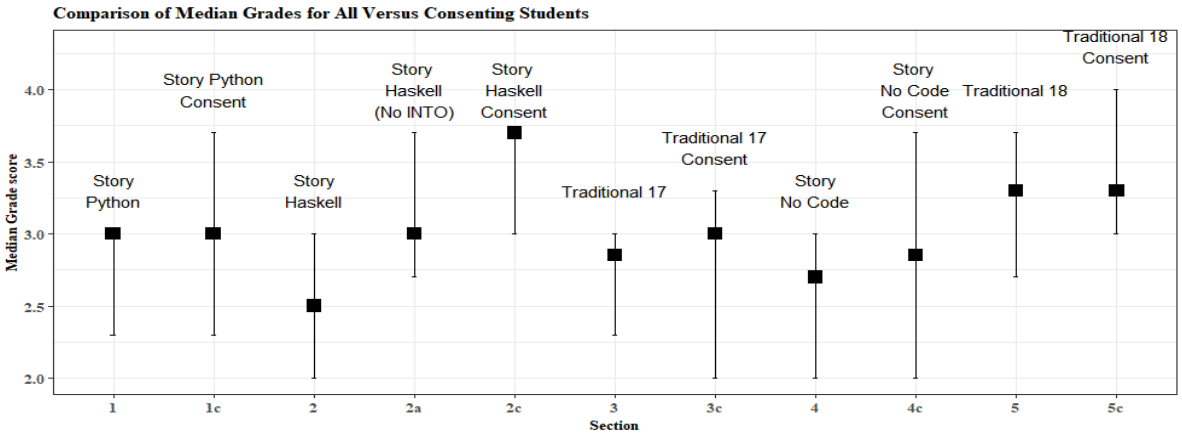

Figure 4: Median CS 1 course grades for all students versus consenting students in each section (including removing the INTO students from the Story Haskell section.

Assignment/Exam	Chi-squared	p-value
Final CS 1 Grade	4.5296	0.339
Assignment 1 (variables/input)	8.5165	0.07439
Assignment 2 (flow of control-if/else)	6.2719	0.1797
Assignment 3 (repetition)	6.4027	0.171
Assignment 4 (functions/decomposition)	8.9108	0.06337
Assignment 5 (1-d arrays)	8.302	0.08112
Assignment 6 (2-d arrays)	1.9485	0.7452
Exam I (Assignment 1-3)	7.3657	0.1178
Exam II (Assignment 4-6)	7.4153	0.1155

Table 3: Difference in grades on CS 1 assignments and exams grouped by CS 0 section.

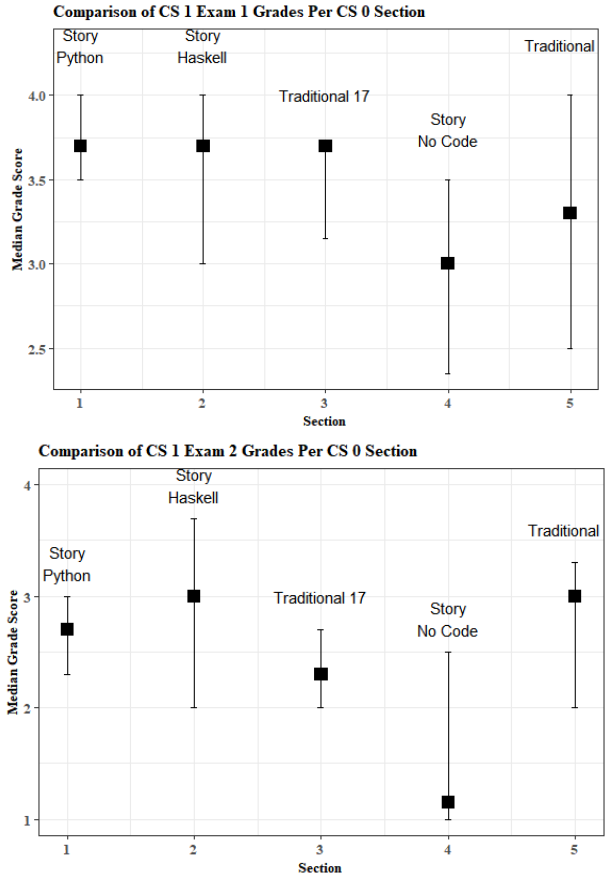

Figure 5: Median CS 1 Exam 1 and 2 grades for consenting students.

5 CONCLUSIONS

Coding is not the only way to orient undergraduate students to the discipline of computer science at a university, and in fact, it might 1) deter/intimidate students from taking the classes, 2) distract and frustrate students with language syntax, and 3) make students without the prior coding experience feel inferior to those with preparatory privilege without providing much advantage in a subsequent C++ programming class. Our research findings align with other research studies indicating that the mere exposure to prior programming is not a strong predictor of success, but yet, the most well-known approach to introducing undergraduate students to computer science is with coding.

Our past research has shown that Story Programming is an alternative for a CS 0 class, and this research study shows that students who receive this introduction perform as well as or better than those that are taught a traditional Python-coding approach. Actually, students from the Story Programming Haskell approach who are not in the international bridge program consistently outperform students from the traditional coding approaches in CS 1. However, students in CS 0 without any coding tend to do a little worse, even though there is no statistical difference in these mean values for grades, assignments, and exams. This suggests that the approach used and amount of coding in CS 0 does not impact students' grades in CS 1, as much as making below a B in CS 0 does. This is the motivation for using a more abstract approach focused on algorithmic concepts, such as the Story Programming approach, as a more inclusive alternative to the traditional code-first approach.

Based on the results from this study, we plan to offer the Story Programming with Haskell again to get another comparison group, and we will likely not offer another Story Programming section without coding. While this approach without coding might be a good option for students outside the major or not continuing to the subsequent C++ programming class, it does not seem to prepare the students as well as the classes with some exposure to coding. In the future, we plan to continue to investigate the impacts different orientations have on class performance beyond CS 1, such as CS 2 and Data Structures.

REFERENCES

[1] T. Bell, J. Alexander, I. Freeman, and M. Grimley. 2009. Computer Science Unplugged: school students doing real computing without computers. *Journal of Applied Computing and Information Technology 13*, 1.

[2] T. Bell, I. H. Witten, M. Fellows, and R. Adams. 2002. Computer science unplugged. An enrichment and extension programme for primary-aged children. Canterbury.

[3] T. J. Cortina. 2015. Reaching a broader population of students through "unplugged" activities. *Communications of the ACM,* 58(3):25-27.

[4] Paul Curzon. 2013. cs4fn and computational thinking unplugged. In *Proceedings of the 8th Workshop in Primary and Secondary Computing Education* (WiPSE '13). ACM, New York, NY, USA, 47-50. DOI: http://doi.acm.org/10.1145/2532748.2611263.

[5] Holger Danielsiek and Jan Vahrenhold. 2016. Stay on These Roads: Potential Factors Indicating Students' Performance in a CS2 Course. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (SIGCSE '16). ACM, New York, NY, USA, 12-17. DOI: https://doi.org/10.1145/2839509.2844591.

[6] Martin Erwig. 2017. *Once Upon an Algorithm.* MIT Press.

[7] Peter B. Henderson. 2011. Computing unplugged enrichment. *ACM Inroads 2*, 3 (August 2011), 24-25. DOI: http://dx.doi.org/10.1145/ 2003616.2003626.

[8] Felienne Hermans and Efthimia Aivaloglou. 2017. To Scratch or not to Scratch?: A controlled experiment comparing plugged first and unplugged

first programming lessons. In *Proceedings of the 12th Workshop on Primary and Secondary Computing Education* (WiPSCE '17), Erik Barendsen and Peter Hubwieser (Eds.). ACM, New York, NY, USA, 49-56. DOI: https://doi.org/10.1145/3137065.3137072.

[9] Michael Haungs, Christopher Clark, John Clements, and David Janzen. 2012. Improving first-year success and retention through interest-based CS0 courses. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education* (SIGCSE '12). ACM, New York, NY, USA, 589-594. DOI=http://dx.doi.org/10.1145/2157136.2157307.

[10] Dennis Kafura and Deborah Tatar. 2011. Initial experience with a computational thinking course for computer science students. In *Proceedings of the 42nd ACM technical symposium on Computerscience education* (SIGCSE '11). ACM, New York, NY, USA, 251-256. DOI: http://dx.doi.org/10.1145/1953163.1953242

[11] Michael S. Kirkpatrick and Chris Mayfield. 2017. Evaluating an Alternative CS1 for Students with Prior Programming Experience. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (SIGCSE '17). ACM, New York, NY, USA, 333-338. DOI: https://doi.org/10.1145/3017680.3017759

[12] Lynn Lambert. 2015. Factors that predict success in CS1. *J. Comput. Sci. Coll.* 31, 2 (December 2015), 165-171.

[13] David J. Malan. 2010. Reinventing CS50. In *Proceedings of the 41st ACM technical symposium on Computer science education* (SIGCSE '10). ACM, New York, NY, USA, 152-156. DOI: http://dx.doi.org/10.1145/1734263.1734316

[14] Cindy Marling and David Juedes. 2016. CS0 for Computer Science Majors at Ohio University. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (SIGCSE '16). ACM, New York, NY, USA, 138-143. DOI: https://doi.org/10.1145/2839509.2844624

[15] Bruce A. Maxwell and Stephanie R. Taylor. 2017. Comparing Outcomes Across Different Contexts in CS1. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (SIGCSE '17). ACM, New York, NY, USA, 399-403. DOI: https://doi.org/10.1145/3017680.3017757

[16] Jennifer Parham-Mocello, Shannon Ernst, Martin Erwig, Lily Shellhammer, and Emily Dominguez. 2019. Story Programming: Explaining Computer Science Before Coding. *ACM Int. Symp. on Computer Science Education* (*SIGCSE 2019).*

[17] Brandon Rodriguez, Cyndi Rader, and Tracy Camp. 2016. Using Student Performance to Assess CS Unplugged Activities in a Classroom Environment. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education* (ITiCSE '16). ACM, New York, NY, USA, 95100. DOI: https://doi.org/10.1145/2899415.2899465

[18] Nathan Rountree, Janet Rountree, and Anthony Robins. 2002. Predictors of success and failure in a CS1 course. *SIGCSE Bull.* 34, 4 (December 2002), 121-124. DOI=http://dx.doi.org/10.1145/820127.820182

[19] Nathan Rountree, Janet Rountree, Anthony Robins, and Robert Hannah. 2004. Interacting factors that predict success and failure in a CS1 course. *SIGCSE Bull.* 36, 4 (June 2004), 101-104. DOI: https://doi.org/10.1145/1041624.1041669

[20] David H. Smith, IV, Qiang Hao, Filip Jagodzinski, Yan Liu, and Vishal Gupta. 2019. Quantifying the Effects of Prior Knowledge in Entry-Level Programming Courses. In *Proceedings of the ACM Conference on Global Computing Education* (CompEd '19). ACM, New York, NY, USA, 30-36. DOI: https://doi.org/10.1145/3300115.3309503

[21] Renate Thies and Jan Vahrenhold. 2013. On plugging "unplugged" into CS classes. In *Proceeding of the 44th ACM technical symposium on Computer science education* (SIGCSE '13). ACM, New York, NY, USA, 365-370. DOI: http://dx.doi.org/10.1145/2445196.2445303

[22] Michele Van Dyne and Jeffrey Braun. 2014. Effectiveness of a computational thinking (CS0) course on student analytical skills. In *Proceedings of the 45th ACM technical symposium on Computer science education* (SIGCSE '14). ACM, New York, NY, USA, 133-138. DOI: http://dx.doi.org/10.1145/2538862.2538956

[23] Brenda Cantwell Wilson and Sharon Shrock. 2001. Contributing to success in an introductory computer science course: a study of twelve factors. *SIGCSE Bull.* 33, 1 (February 2001), 184-188. DOI: https://doi.org/10.1145/366413.364581

[24] Zoë J. Wood, John Clements, Zachary Peterson, David Janzen, Hugh Smith, Michael Haungs, Julie Workman, John Bellardo, and Bruce DeBruhl. 2018. Mixed Approaches to CS0: Exploring Topic and Pedagogy Variance after Six Years of CS0. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (SIGCSE '18). ACM, New York, NY, USA, 20-25. DOI: https://doi.org/10.1145/3159450.3159592

.