JCSI
Journal of Computer Science Integration

Volume 2 | Issue 2 Article 1

2019

Coding Success through Math Intervention in an Elementary School in Rural Amish Country

Megan Brannon
Kent State University

Elena Novak
Kent State University

Recommended Citation

Brannon, M., & Novak, E. (2019). Coding Success through Math Intervention in an Elementary School in Rural Amish Country. *Journal of Computer Science Integration, 2* (2), 1-10. https://doi.org/10.26716/jcsi.2019.02.2.1

JCSI
Journal of Computer Science Integration

Coding Success through Math Intervention in an Elementary School in Rural Amish Country

Abstract

Coding in the elementary classroom is a relatively new movement in K-12 education that intends to engage young people in computer science and technology-related study. Coding initiatives focus on introducing young learners to coding and developing their computational thinking abilities. Coding helps enhance problem solving, mathematics skills, and higher-order thinking. Nevertheless, educators face many challenges with teaching coding at the elementary school level, because of the newness of computer science concepts and programming languages, gaps in student mathematics knowledge, use of technology, a relatively short attention span of young students and not fully developed reasoning, logic, and inferential skills among many others. This report describes how math interventions helped elementary school students in rural Amish Country become more successful with their coding activities.

Keywords

coding, mathematics, elementary school

DOI

10.26716/jcsi.2019.02.2.1

Corresponding Author

Megan Brannon
Kent State University
School of Teaching, Learning and Curriculum Studies
150 Terrace Drive
P.O. Box 5190
Kent, OH 44242-0001

Coding and Computational Thinking

In recent years, coding programs such as Code.org, Tynker, Scratch, and many others have made coding more accessible than ever to elementary classrooms (Davis, 2013; Kumar, 2014; Grover & Pea, 2013). There are also toys and games that promote coding skills in fun, interactive ways for students. However, just because coding is becoming more accessible to the classroom does not mean that it is without frustration for students who engage in classroom coding activities.

Learning to code presents many benefits. It helps students develop better problem-solving abilities and enhance their higher order thinking skills (Falloon, 2016). In addition, coding engages students in genuine situations in which skills such as mathematics, algorithms, problem solving, and collective analysis can occur (Isreal et al., 2015; Fessakis, Gouli, & Mavroudi, 2013; Jona et al., 2014). The relationship between coding and mathematics is not surprising, as coding, also known as computer science or programming, has its roots in mathematics. Byrne and Lyons (2001) discuss the relationship between mathematics and coding, describing how the skills related to learning programming are comparable to the skills needed to be proficient in mathematics. Additionally, coding skills assist students in becoming better mathematicians. A basic calculator can do arithmetic, just as most students can. However, to think mathematically, students need to understand how mathematics function authentically. Research shows that coding can provide motivation for learning mathematical processes and skills (Calder, 2010). For example, in a study with the Scratch coding program, Calder (2010) found that while Scratch was not originally created to facilitate the growth of mathematical skills, the program clearly did immerse students in mathematical concepts such as positionality, measurement (angle and time), and spatial perception. Calder (2010) also found that math skills were further developed using problem solving, logic, and reasoning.

Nevertheless, learning to code is challenging for many students. Some of these challenges lie in the newness of the material, meaning the new syntax of the language (Ben-Ari, 2001). When learning to code, students learn new vocabulary, new concepts, new programming language, and a completely new way of approaching the material. Malan and Leitner (2007) illustrate learning coding as learning a new language like Greek, with some pieces taken from the English language but with a wholly unfamiliar syntax. Byrne and Lyon (2001) also use a language comparison when they correlate language grammar skills to the syntax of a programming language, stating “attention to construction and syntax [in coding] might be considered similar to language grammar skills, and creative writing skills might be considered similar to developing innovative programming solutions” (p. 52). Some foreign terms that students may have difficulty with include loops, variables, algorithms, Boolean expressions, and conditional logic (Malan & Leitner, 2007). These concepts are alien to most elementary students when they first learn coding.

Another issue with coding relates to the use of the technology. Technology is ubiquitous for students. Most students use devices often both in and out of school. However, this does not mean that students are using devices productively. Kafai and Burke (2013) describe our students,

the digital natives, and explain that they may be able to use technology, but not always for purposes involving creativity, critical thinking, or productivity. Coding fits into both the critical and creative thinking categories.

Coding uses reasoning, logic, and inferential skills, which tend to be much more cognitively involved. According to Piaget, "deductive reasoning is a cognitively advanced skill that develops during adolescence; in particular, adolescents acquire a complete mental logic that corresponds to standard logic" (Chao & Cheng, 2000, p. 40). Because deductive reasoning is not fully evolved until adolescence, many elementary students find it difficult to apply logic and reason to their study of coding. Students who code typically have to work through multi-step activities as well, which can be difficult for their short attention spans (Altun, Hazar & Hazar, 2016). Keeping attention is imperative to success in the coding field, which sometimes requires time-intensive work to solve a problem.

Coding, also known as programming, aids students in learning both problem solving and the design process (e.g. iteration and modularization) which are skills that can also be used outside of the computer science context (Resnick et al., 2009). Computational thinking is sometimes thought of as thinking like a computer; however, its goal is realistically to aid students in constructing the mental processes to effectively use a computer in solving real world problems (Lu & Fletcher, 2009). Wing (2006) describes computational thinking as a collaborative tool that rises above machine knowledge to help people solve problems that they might be unable to solve by themselves. One of the benefits of computational thinking is that it allows learners to think in a more organized, systematic way (Kafai & Burke, 2013). Computational thinking also allows learners to develop mathematical expertise (Sengupta, Kinnebrew, Basu, Biswas, & Clark, 2013). The five steps to computational thinking which aid in problem solving are: decomposition, pattern recognition, abstraction, generalization, and algorithm building. These five steps of computational thinking are supported by the Common Core State Standards (CCSS) Standards for Mathematical Practice:

1. **Problems are broken down into manageable portions** (*decomposition*; CCSS.Math.Practice.MP 1: Make sense of problems and persevere in solving them)
2. **Information is analyzed for patterns** (*pattern recognition*; CCSS.Math.Practice.MP 8: Look for and express regularity in repeated reasoning)
3. **Unnecessary information is removed from the problem** (*abstraction*; CCSS.Math.Practice.MP 6: Attend to precision)
4. **Make the solution work for multiple problems if possible** (*generalization*; CCSS.Math.Practice.MP 2: Reason abstractly and quantitatively)
5. **Step-by-step instructions are generated to create a solution** (*algorithm building*; CCSS.Math.Practice.MP 7: Look for and make use of structure)

Coding Through Math Intervention: Evidence from the Field

This report describes coding activities in an elementary public school in rural Ohio that was situated in the heart of Amish Country. Because of the district's location in Amish Country,

and the presence of local farms, the district serves a large student population with diverse language, cultural understandings, and religious beliefs. Cultures represented in this district included Amish, Mennonite, Latinx, and Anglo. Languages spoken included English, Pennsylvania Dutch, German, Spanish, and several Spanish dialects. The district was also surrounded by a lower socioeconomic area, with school-wide Title I services (math and reading) in most of the elementary buildings.

Before 2015, there had never been a technology education program in this district. When the technology education program began, computer science was the last thing on the district's agenda. For the technology education program, a single technology teacher serviced all of the elementary buildings in the district, on a rotating schedule with approximately 40 minutes per week of instruction with each grade level. With such a huge learning curve for the students, who grew up in mostly conservative Amish/Mennonite households, the district technology curriculum goals were to start slow and teach students about applications of technology in the beginning. The district had to eliminate the cultural fear of technology while still preparing students for online standardized testing and 21st Century Careers. As time moved on, and students became less afraid and more proficient technology users, the technology curriculum was able to become more challenging. In order to make the curriculum more challenging, the district decided to integrate computer science, or coding, into the elementary technology curriculum.

Coding Struggles

Soon after integrating the Code.org Curriculum, a roadblock to integration materialized: some students began to struggle with the coding content. Students possessed varying skill levels in the area of coding. These large variances were puzzling to the district and teachers because all students began coding at the same time and had similar experiences with technology education. Therefore, the district delved deeper into why some students were struggling and some were not. After discussions with teachers and many informal classroom observations, the conclusion arose that many students were struggling with the mathematics portion of the coding program. These mathematics struggles were split into two categories, which coincide with Moursund's (2006) two components of mathematics: mathematics content and mathematics maturity. Mathematics content involves typical mathematics skills (e.g. basic operations, geometry, algebra, etc.). Mathematics maturity involves solving problems, logic and reasoning, and transfer of knowledge from one setting to another.

The National Research Council (2009) has shown that it is not unusual to discover mathematics deficits in students from areas with low socioeconomic status such as this school district. Students of families with a low socioeconomic status tend to score lower on standardized tests than students from a higher socioeconomic background do (Bradley & Corwyn, 2002; Duncan & Magnuson, 2005; Galindo & Sonnenschein, 2015). Additionally, about a third of fourth grade children from lower socioeconomic backgrounds do not meet grade level expectations in mathematics (Bachman et al., 2015; U.S. Department of Education, 2005). There is also an increasing need for more critical thinking in mathematics class. Elementary students

from lower socioeconomic backgrounds tend to receive basic instruction on mathematics skills and procedures compared to their peers of higher socioeconomic status who are often encouraged to engage in high-level thinking activities (Bachman et al., 2015; Desimone & Long, 2010; Means & Knapp, 1991). Coding directly involves these higher order-thinking skills.

Because these mathematics and problem solving challenges directly affected student classroom performance, something had to be done to help the students succeed in coding. Thus, instead of suggesting tutoring to students' parents, or placing more on the classroom or intervention teachers, math intervention groups were formed and utilized during the technology class time. Essentially, the goal was to improve coding performance through math intervention. The Code.org tracking system, informal classroom observations, and input/data from classroom teachers were utilized to select students who needed help with mathematics skill and problem solving. The classroom teachers were extremely helpful in creating content-standards based interventions.

In order to keep students working in the technology classroom while small group interventions were happening, the *Ask 3, then Me* rule was adopted, which allowed students to ask others in the classroom for help in order to create uninterrupted intervention time. The students were also allowed to participate in paired programming every other week. Paired programming is a system where students code using a partner. This type of peer mentoring is very effective in the coding classroom, and it acts as its own additional intervention when students have difficulties in certain areas, such as mathematics.

As stated before, students who struggled fell into two distinct mathematics groups, i.e., content and maturity. The students who struggled with mathematics content knowledge typically had trouble with the following concepts in Code.org:

- Geometric shapes
- Measuring sides of shapes
- Measuring and understanding angles
- The four major operations (+, -, *, /)
- Using digital tools (such as a protractor)
- Basic algebra (finding an unknown/variables)
- Understanding and extending patterns
- Using a coordinate grid

As an example, in one Code.org activity, students must create a circular pattern by repeatedly drawing circles and turning (Figure 1). For students to correctly finish the pattern, and the Code.org task, they must possess knowledge of directionality (turning right or left), measurement (moving a certain number of pixels), and measuring angles (turning the correct amount of degrees).

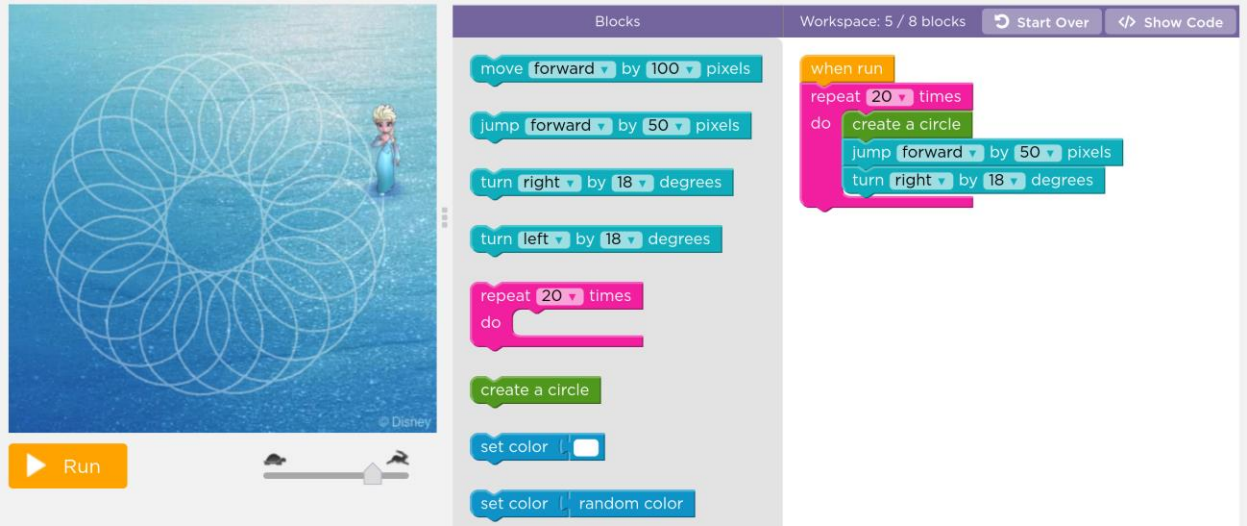

Figure 1. This activity uses Disney's Frozen character Elsa to help students draw a circular pattern. Students place code on the right that involves turning the right number of degrees and then continuing in order to draw the correct pattern.

Other coding activities involved students moving around one square at a time (Figure 2). These types of activities, for lower elementary students, involve the use of the repeat block (looping). This type of activity involves students in moving around a grid as well as adding or multiplying (looping the action a correct number of times). Pattern recognition and extension are also important in tasks such as this one. Because students need to complete the task with a limited number of programming blocks, they must recognize patterns in order to use the repeat block in place of multiple other blocks.

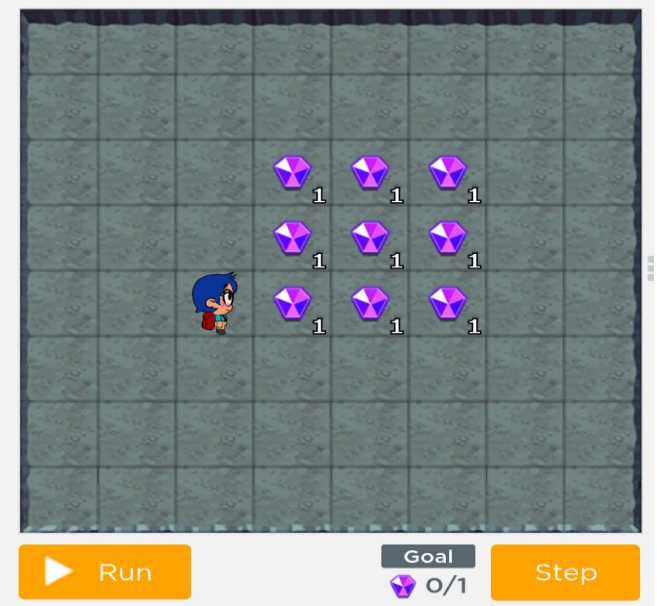

Figure 2. The user can use an understanding of pattern building and extension as well as multiplication or addition to successfully place repeat blocks in levels such as this one.

Even some of the more difficult coding levels involve basic arithmetic. Figure 3 presents a coding sample that involves subtracting multiple numbers from a total (counter). Students completing this task also work with variables in an authentic format.

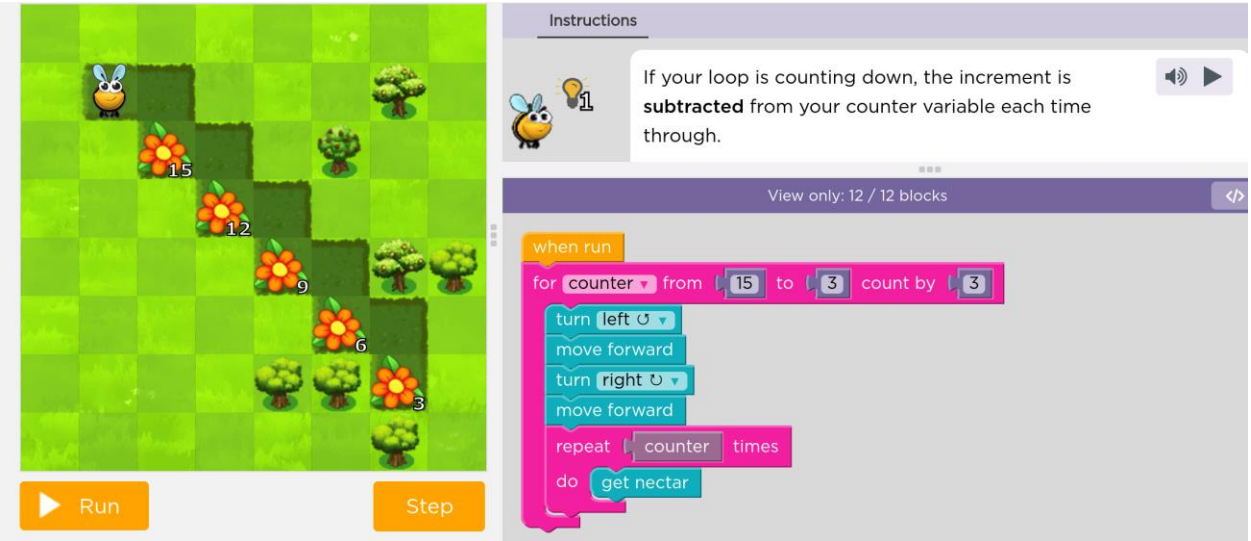

Figure 3. Basic arithmetic operations can be used in levels such as the one above.

All the mathematics skills presented in the above coding activities are part of the CCSS standards for mathematical practice at elementary grade levels. Students in Grades 3-6 had the most difficulty with these basic mathematics skills. Interventions with this group involved explicit instruction on the skills with which students struggled. Materials such as worksheets, interactive online learning games, and dry erase boards were typical for interventions with the students in this group.

The other group of students, those who struggled with mathematics maturity, were not concentrated to one specific grade band. Interestingly, students in this second group tended to be some of the higher achieving students in the regular classroom, yet they struggled with coding. The specific skills this group of learners struggled with included higher order thinking, multi-step problems, and tasks which required deeper thinking. Problem-solving interventions included explicit teaching of problem-solving strategies/computational thinking skills, reading children's literature about problem solving, and practicing rigorous, multi-step problems. Since coding can be conceptually challenging, the goal with this group was to teach the students to think like coders and use the concepts of computational thinking. Boosting self-esteem was also important with this group because, in many cases, one of their setbacks was a fear of failure. These students needed to learn how to fail in order to succeed, and children's literature was very helpful in this aspect. Learning about growth mindsets and encouraging them in students was also very helpful in building their confidence. Some of the literature examined during these small group sessions included:

- *Rosie Revere, Engineer* Written by Andrea Beaty and Illustrated by David Roberts
- *Beautiful Oops!* Written and Illustrated by Barney Saltzberg
- *What Do You Do with a Problem?* Written by Kobi Yamada and Mae Besom
- *What Do You Do with an Idea?* Written by Kobi Yamada and Mae Besom
- *What Do You Do with a Chance?* Written by Kobi Yamada and Mae Besom
- *The Most Magnificent Thing* Written by Ashley Spires
- *The Girl Who Never Made Mistakes* Written by Mark Pett and Gary Rubinstein
- *I Can't Do That, YET: Growth Mindset* Written by Esther Pia Cordova and Illustrated by Maima W Adiputri

After approximately two months of math intervention, there were marked improvements in students' coding skills. The fact that math intervention helped the struggling students in computer science class suggests how integral mathematics is to the study of computer science in the elementary classroom. Not only were struggling students more proficient with the coding program after math intervention, they also displayed markedly improved confidence in the area of coding. After intervention, students appeared to have more fun and be less stressed when completing the online coding lessons.

Conclusion

Mathematics and coding can be viewed as two islands across from each other, close together yet not touching, and the concept of computational thinking is the bridge between these

two lands. Teaching both mathematics and coding concomitantly helps to develop a strong mathematics AND coding skills. As teachers, we must work for our students to help eliminate deficiencies in learning. This is just as true for the computer science classroom as it is for the mathematics classroom. While math tutoring may not be seen as a typical intervention, in this case it was effective in improving computer science classroom performance. Coding programs are just a tool, and they are not a substitute for effective teaching methods or the interventions described here. By the end of the year in this elementary school, students of different cultures, students of different socioeconomic backgrounds, and students who speak different languages, with the help of some math and problem-solving intervention for some, were all able to speak the same language: coding.

About the Authors

Megan Brannon is a student in the Curriculum and Instruction- Educational Technology doctoral program at Kent State University. She has more than ten years of experience in K-12 education, working as both a teacher and technology integration specialist for her school district. Through her K-6 technology experience, she helped to develop the district's technology curriculum and helped bring computer science to the district for the first time, which also included carrying out the math interventions in the case study above. Megan's research interests include best practices in technology integration, computational thinking, and viewing computer science from various perspectives (college preparation, math, ELL students, and teacher professional development).

Elena Novak is an Associate Professor of Educational Technology at Kent State University. She earned her Ph.D. in Instructional Systems and Learning Technologies from Florida State University. Elena Novak's research aims to advance STEM Education using learning technologies.

References

Altun, M., Hazar, M., & Hazar, Z. (2016). Investigation of the Effects of Brain Teasers on Attention Spans of Pre-School Children. *International Journal of Environmental and Science Education, 11*(15), 8112-8119.

Bachman, H. J., Votruba-Drzal, E., El Nokali, N. E., & Castle Heatly, M. (2015). Opportunities for learning math in elementary school: Implications for SES disparities in procedural and conceptual math skills. *American Educational Research Journal, 52*(5), 894-923. https://doi.org/10.3102/0002831215594877

Ben-Ari, M. (2001). Constructivism in computer science education. *Journal of Computers in Mathematics and Science Teaching, 20*(1), 45-73.

Bradley, R. H., & Corwyn, R. F. (2003). Age and ethnic variations in family process mediators of SES.

Byrne, P., & Lyons, G. (2001). The effect of student attributes on success in programming. *Acm sigcse bulletin, 33*(3), 49-52. https://doi.org/10.1145/507758.377467

Calder, N. (2010). Using Scratch: an integrated problem-solving approach to mathematical thinking. *Australian Primary Mathematics Classroom, 15*(4), 9–14.

Chao, S. J., & Cheng, P. W. (2000). The emergence of inferential rules: The use of pragmatic reasoning schemas by preschoolers. *Cognitive Development, 15*(1), 39-62.

Davis, M. R. (2013). Computer coding lessons expanding for k-12 students. *Education Week*.

Desimone, L. M., & Long, D. (2010). Teacher effects and the achievement gap: Do teacher and teaching quality influence the achievement gap between Black and White and high- and low-SES students in the early grades? *Teachers College, 112*, 3024–3073.

Duncan, G. J. & Magnuson, K. A. (2005). Can Family Socioeconomic Resources Account for Racial and Ethnic Test Score Gaps? *The Future of Children* 15(1), 35-54. Princeton University. Retrieved September 5, 2019, from Project MUSE database. https://doi.org/10.1353/foc.2005.0004

Falloon, G. (2016). An analysis of young students' thinking when completing basic coding tasks using Scratch Jnr. On the iPad. *Journal of Computer Assisted Learning, 32*(6), 576-593. https://doi.org/10.1111/jcal.12155

Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5-6 years old kindergarten children in a computer programming environment: A case study. *Computers & Education*, 63, 87-97. https://doi.org/10.1016/j.compedu.2012.11.016

Galindo, C., & Sonnenschein, S. (2015). Decreasing the SES math achievement gap: Initial math proficiency and home learning environments. *Contemporary Educational Psychology, 43*, 25-38. https://doi.org/10.1016/j.cedpsych.2015.08.003

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational Researcher*, *42*(1), 38-43. https://doi.org/10.3102/0013189x12463051

Jona, K., Wilensky, U., Trouille, L., Horn, M., Orton, K., Weintrop, D., & Beheshti, E. (2014, January). *Embedding computational thinking in science, technology, engineering, and math (CT-STEM)*. Paper presented at the Future Directions in Computer Science Education Summit Meeting, Orlando, FL. https://doi.org/10.1007/978-3-319-08189-2_3

Kafai, Y. B., & Burke, Q. (2013). Computer programming goes back to school. *Phi Delta Kappan*, *95*(1), 61-65. https://doi.org/10.1177/003172171309500111

Kumar, D. (2014). Digital playgrounds for early computing education. *ACM Inroads*, *5*(1), 20-21. https://doi.org/10.1145/2568195.2568200

Lu, J. J., & Fletcher, G. H. (2009, March). Thinking about computational thinking. In *ACM SIGCSE Bulletin, 41(1),* 260-26. https://doi.org/10.1145/1539024.1508959

Malan, D. J., & Leitner, H. H. (2007). Scratch for budding computer scientists. *ACM Sigcse Bulletin*, *39*(1), 223-227. https://doi.org/10.1145/1227504.1227388

Means, B., & Knapp, M. S. (1991). Cognitive approaches to teaching advanced skills to educationally disadvantaged students. Phi Delta Kappan, 73, 282–289.

Moursund, D. G. (2006). *Computational thinking and math maturity: Improving math education in K-8 schools*.

National Research Council. (2009). *Mathematics learning in early childhood: Paths toward excellence and equity*. National Academies Press.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., ... & Kafai, Y. B. (2009). Scratch: Programming for all. *Commun. Acm*, *52*(11), 60-67. https://doi.org/10.1145/1592761.1592779

Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating computational thinking with K-12 science education using agent-based computation: A theoretical framework. *Education and Information Technologies, 18*(2), 351-380. https://doi.org/10.1007/s10639-012-9240-x

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33-35.

U.S. Department of Education. (2005). Nation's report card: Mathematics 2005. Washington, DC: National Center for Education Statistics.