knowledge

MDPI

Article

The Impact of a Computing Curriculum Accessible to Students with ASD on the Development of Computing Artifacts

Abdu Arslanyilmaz [1,*], Margaret L. Briley [1], Gregory V. Boerio [1], Katie Petridis [2], Ramlah Ilyas [1] and Feng Yu [1]

[1] School of Computer Science, Information, and Engineering Technology, Younstown State University, Youngstown, OH 44555, USA; fyu@ysu.edu (F.Y.)
[2] Potential Development Program, Youngstown, OH 44507, USA
* Correspondence: aarslanyilmaz@ysu.edu; Tel.: +1-330-941-3120

Abstract: There has been no study examining the effectiveness of an accessible computing curriculum for students with autism spectrum disorder (ASD) on their learning of computational thinking concepts (CTCs), flow control, data representation, abstraction, user interactivity, synchronization, parallelism, and logic. This study aims to investigate the effects of an accessible computing curriculum for students with ASD on their learning of CTCs as measured by the scores of 312 computing artifacts developed by two groups of students with ASD. Conducted among 21 seventh-grade students with ASD (10 in the experimental group and 11 in the control), this study involved collecting data on the computing projects of these students over 24 instructional sessions. Group classification was considered the independent variable, and computing project scores were set as the dependent variables. The results showed that the original curriculum was statistically significantly more effective for students in learning logic than the accessible one when all seven CTCs were examined as a single construct. Both curriculums were statistically significantly effective in progressively improving students' learning of data representation, abstraction, synchronization, parallelism, and all CTCs as a single construct when examining the gradual increase in their computing artifact scores over the 24 sessions. Both curriculums were statistically significantly effective in increasing the scores of synchronization and all CTCs as a single construct when the correlations between CTCs and sessions for individual groups were analyzed. The findings underscore that students with ASD can effectively learn computing skills through accessible or standard curriculums, provided that adjustments are made during delivery.

Keywords: computational thinking; computational thinking concepts; autism spectrum disorders; accessible computing curriculum

Citation: Arslanyilmaz, A.; Briley, M.L.; Boerio, G.V.; Petridis, K.; Ilyas, R.; Yu, F. The Impact of a Computing Curriculum Accessible to Students with ASD on the Development of Computing Artifacts. *Knowledge* **2024**, *4*, 85–95. https://doi.org/10.3390/knowledge4010005

Academic Editor: Constantin Bratianu

Received: 21 December 2023
Revised: 26 February 2024
Accepted: 29 February 2024
Published: 5 March 2024

1. Introduction

Computational thinking (CT) is a crucial skill in the 21st century and will be integral to driving new discoveries and innovations across diverse fields and endeavors. Accordingly, many research studies have been conducted to examine the effects of CT on various aspects of learning, teaching, and cognition at different levels.

Research indicates that CT instruction enhances students' interest, knowledge, and skills in computing [1–5]; improves quantitative and critical thinking abilities [6]; develops skills in abstracting, generalizing, and writing persuasive arguments [7]; facilitates the transfer of problem-solving skills in both near and far contexts while enhancing spatial relations abilities [8]; aids in comprehending algorithmic flow [9]; and shows potential in predicting future academic success [10,11].

Additionally, some reports indicated that proficiency in CT during college freshman courses can serve as an indicator for future academic success [10], and a strong correlation exists between students' computational skills and their academic performance [11]. In addition, CT has the potential to enhance both quantitative and critical thinking skills [6]

and can assist in abstracting information, making generalizations, and crafting persuasive arguments [7], and a curriculum centered around CT enhances students' perceptions of, interest in, and knowledge about computing [3,5]. Also, research has indicated that coding contributes to both short- and long-term transfer of problem-solving skills, as well as an improvement in the ability to solve spatial relations, particularly among fifth- and sixth-grade students [8]. Also, integrating CT into middle school and high school classes has been shown to increase knowledge in algorithmic flow [9] and middle school students' CT skills [1].

Furthermore, significant correlations between computational thinking and critical thinking, as well as pre-service teachers' modeling skills, were reported in [12,13], where it was found that the fundamental concepts of CT promoted, organized, and structured English writing; increased students' writing motivation; and reduced their writing anxiety. Ref. [14] reported that students' understanding and application of linear algebra topics were improved by familiarity with essential computational thinking concepts. Ref. [15] analyzed the effect of an intervention program in computational thinking on the executive functions of school-age children and reported that the intervention had significant effects on the executive functions associated with the anterior prefrontal cortex and the dorsolateral cortex. Ref. [16] reported that computational thinking has a positive impact on both digital self-efficacy and self-exploration. Ref. [17] investigated the relationship between middle school students' computational thinking skills and their STEM career interest and attitudes toward inquiry, and a significant correlation was observed between middle school students' computational thinking skills and their STEM career interest and attitudes toward inquiry.

Consequently, CT should be taught to all students alongside fundamental skills like reading, writing, and arithmetic [18]. Accordingly, researchers have studied and demonstrated the successful integration of CT into various subjects across different grade levels for mainstream students [3–11].

For example, the authors of [19] examined the integration of CT into mathematics education, focusing on classroom practices and gathering opinions from students and pre-service teachers regarding unplugged CS activities. Similarly, the authors of [16] provided a framework for social studies teachers regarding the integration of CT into K-12 classrooms. Another study explored the incorporation of CT into English writing instruction [13]. In addition, in [20], the authors designed and implemented computational thinking activities for mathematics students. Finally, the authors of [14] developed curriculum materials to introduce students to computational thinking concepts within linear algebra topics.

Although many studies have focused on creating and implementing CT curricula for mainstream students, there has been a limited amount of research on designing and developing CT curricula specifically for students with special needs. Among these rare studies is the work by [21] aimed at high school students diagnosed with learning disabilities or attention deficit disorder (commonly known as ADHD). This study identified challenges in teaching a computer science (CS) course to students with ADHD, proposing and testing adjustments to create an accessible CS course for them. Successful adjustments were reported, particularly addressing the barriers related to language, reading, written expression, math, and attention [21]. Another rare study was conducted by [22], where adaptations and accommodations were developed to make an original CT curriculum accessible to students with ASD.

However, there have not been a sufficient number of studies focused on a thorough examination of an accessible CT curriculum's effectiveness in improving learning outcomes for students with ASD. One such study, titled "Assessing the Efficacy of an Accessible Computing Curriculum for Students with ASD", evaluated the effectiveness of an accessible CT curriculum by analyzing pretest and post-test scores, as well as computational thinking processes, through artifact-based interview scores. According to the study results, students with ASD demonstrated improved computational thinking concepts when exposed to a computing-based curriculum, whether accessible or not. Specifically, the accessible computing curriculum showed significant improvements in students' proficiency

in debugging and testing, iterating and experimenting, modularizing and abstracting, and remixing and reusing. However, relying solely on pretest, post-test, and interview scores does not provide a comprehensive understanding of progressive learning, improvements, and the development of computational thinking concepts throughout the duration of the intervention. They may offer an objective way of assessing single points in time (before and after an intervention), but they may not accurately describe the learning experiences students undergo while developing computing projects over a period of time as they learn computing, which is crucial to measure for a more comprehensive assessment of their learning of computational thinking concepts. These computing projects offer valuable insights into the technical progress they make and the challenges they encounter as they learn CT concepts.

Therefore, this research investigates the effectiveness of an accessible CT curriculum in enhancing students with ASDs' understanding of CTCs (computational thinking concepts: flow control, data representation, abstraction, user interactivity, synchronization, parallelism, and logic) as revealed by the computing projects that they develop. The research involves comparing computing projects created by two sets of students: the experimental group taught with an accessible CT curriculum and the control group taught with the original CT curriculum.

The Problem and Research Question

No study has been conducted on the analysis of computing projects developed by students with ASD to assess the effectiveness of an accessible CT curriculum in improving the learning of CTCs (flow control, data representation, abstraction, user interactivity, synchronization, parallelism, and logic). The research question in this study is as follows:

Is an accessible CT curriculum more effective in enhancing the learning of CTCs among students than a CT curriculum designed for mainstream students?

This research question was explored by conducting a MANOVA analysis on the computing project scores achieved by two groups of students diagnosed with ASD.

2. Participants

Twenty-one seventh-grade students, at an average age of 13, with ASD from two inner-city schools located in the Midwest United States, participated in this research study and were divided into experimental and control groups. All seventh-grade students volunteered to participate in this study. The experimental group in one school was exposed to an accessible computing curriculum, while the control group in the other school attended the sessions under an original CT curriculum. The students were able to use applications on Apple iPads. They had not been exposed to any CT-related instructional activities before the study. Each student was provided with an iPad to carry out the instructional activities in this study, including the utilization of the Scratch programming interface. Students in both groups were equal in their knowledge of CTCs at the beginning of the study, as indicated by the analysis of a pretest using an independent sample *t*-test ($t(18) = 1.89$, two-sided $p = 0.072$). The independent variable comprised two groups: ten students in the experimental group and eleven students in the control group. The dependent variable encompassed the scores of the computing projects developed by these students, assessing their comprehension of CTCs.

3. Materials and Research Methods

3.1. The Original and Accessible CT Curriculums

The original CT curriculum utilized in this study is the creative computing curriculum [23], which was not specifically designed and developed for students with ASD. The creative computing curriculum [23,24] comprises 7 units, with each unit having 6 sessions, totaling 42 sessions overall. It provides instructional activities (a description of the instructional activities, the list of instructional resources, learning objectives, discussion questions,

and questions for evaluating student work) centered around Scratch [25], a simple, visual, block-based programming environment for students of all ages.

The accessible CT curriculum utilized in this study is the one modified and made accessible for students with ASD in a previous study [22] based on the original CT curriculum, and the instructional activities teaching computing in this curriculum also revolved around Scratch [25]. The accessible computing curriculum has a total of 6 units of 6 sessions, for 36 sessions altogether. Only 24 out of the 36 sessions were implemented in this project.

Each session of the accessible CT curriculum was composed of eight key instructional elements. The first one was the session schedule, which was presented on separate pages for easy printing and placement on classroom walls and student desks. Task timings and breaks were personalized based on individual attention spans and behavioral needs. The second involved pre-teaching activities, which comprised three components—topics, terms, and expectations. Unfamiliar terms were explained with descriptions and visual symbols at a level suitable for students' reading abilities. Clear guidance on session expectations helped to alleviate anxiety and prepared students for activities. The third one was session learning objectives, which were presented in two sets—one for informing teachers about session targets and the other for students regarding achievable goals by the session's end. The objectives were designed to be measurable, attainable, and observable, accommodating visual, oral, and written comprehension and responses. Additional objectives were added or removed as required. Instructional activities were simplified and broken down into smaller, manageable sections to accommodate diverse student characteristics (visual/verbal/kinesthetic, individual/group work, various response modes, etc.). Modeling activities were integrated to aid students in following along with teachers' instructions and provided options for independent study. The fourth one included visual handouts, which were developed as step-by-step visual guides (totaling 24) to assist students in navigating session tasks effectively. The fifth one involved instructional videos. Approximately 60 individual instructional videos were created for students preferring visual learning methods, supplementing the instructional content [26]. The sixth one was reflection prompts, which were provided in both verbal and visual formats within each session, allowing students to reflect on the covered topics. The seventh component involved work evaluation rubrics that were created for each session, aligning with learning objectives and designed to be objective, observable, and measurable. The rubrics assessed student achievement across three levels for each item: with physical assistance, with verbal/visual cues, and independently. These adaptations aimed to enhance accessibility, understanding, and inclusivity in the curriculum, catering to various learning preferences and needs among students.

3.2. Methodology

In this study, a total of twenty-four instructional sessions comprising four units were covered for both control and experimental groups (see Figure 1). Each instructional session was taught over two 45 min classes on two separate days of the week, spanning a period of 27 weeks from September 2021 to the end of April 2022 for both control and experimental groups. The duration of the intervention was the same for both groups. While the control group was exposed to the original curriculum, consisting of 4 units of 24 sessions in total, the experimental group had the corresponding 4 units of 24 sessions from the experimental accessible computing curriculum (see Table 1 for the units and topics covered over the 27 weeks). Two full-time teachers facilitated the implementation of the original curriculum, and two full-time teachers facilitated the implementation of the accessible CT curriculum. None of the four teachers involved had any experience teaching computing to students, including students with ASDs.

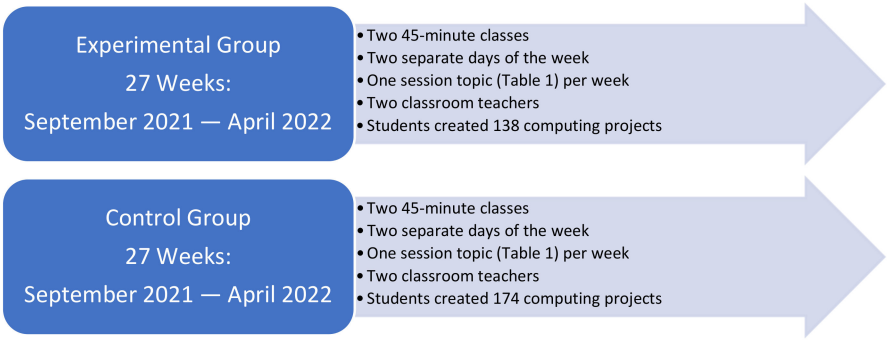

Figure 1. Visual representation of the experiment sessions as outlined in Table 1.

Table 1. Units and session topics covered in both accessible and original CT curriculums.

Unit	Session Topics
1	Introducing Scratch, Scratch Accounts, Design Journal, Scratch Surprise, Scratch Studio, Critique Group
2	Programmed to Dance, Step by Step, Ten Blocks, My Studio, Debug It, About Me
3	Performing Scripts, Build-A-Band, Orange Square—Purple Circle, It's Alive, Debug It!, Music Video
4	Characters, Conversations, Scenes, Debug It!, Create Construction, Pass It On

3.3. Analysis of Data

The analysis and scoring of students' computing projects was carried out using Dr. Scratch, a plugin inspired by Scratch and built on Hairball [27,28]. Dr. Scratch automatically evaluates Scratch projects, analyzing the development of CTCs and identifying certain bad programming practices. This specific plugin was selected because it provides insights into students' application of CTCs learned, encompassing aspects such as flow control, data representation, abstraction, user interactivity, synchronization, parallelism, and logic. Dr. Scratch assigns a score ranging from 0 to 3 points for each concept, culminating in an overall score. These results were utilized to compare the two groups and determine whether the adapted curriculum had any impact on students' comprehension of CTCs.

Both experimental and control groups' computing projects were analyzed to determine the effectiveness of the accessible CT curriculum against the original one in improving students' learning of the CTCs (flow control, data representation, abstraction, user interactivity, synchronization, parallelism, and logic). Pearson product moment correlation analyses were carried out to examine the gradual change in students' computing projects from the first to the last instructional session for both groups.

To determine if the experiment affected students' learning of CTCs, as revealed in their computing projects, a multivariate analysis (MANOVA) was conducted. The dependent variables were the CTCs exhibited on students' computing projects. MANOVA analyzed the independent variable over each dependent variable individually and all together as a combined construct. The level of significance (alpha) was set at 0.05. Wilk's Lambda was used to determine the overall multivariate significance of the dependent variables on the groups. Furthermore, to determine whether the accessible CT curriculum had any impact on students' learning of individual CTCs (flow control, data representation, abstraction, user interactivity, synchronization, parallelism, and logic), 2 × 2 ANOVAs were run separately with the experimental and control groups as the independent variables and each CTC as the dependent variable.

4. Results

4.1. Computing Project Scores

As seen in Table 2, students in the experimental group created a total of 138 projects, while students in the control group created a total of 174 projects. Descriptive statistics (see Table 2) showed that students in the experimental group had higher mean CTC scores on flow control (1.74 vs. 1.55), synchronization (1.04 vs. 1.03), and parallelism (0.747 vs.

0.640) than students in the control group. On the other hand, students in the control group had higher mean CTC scores on data representation (0.918 vs. 0.865), abstraction (0.913 vs. 0.813), user interactivity (1.35 vs. 1.22), and logic (0.199 vs. 0.023).

Table 2. Descriptive statistics on CTC scores of computing artifacts.

	School	Mean	Std. Deviation	N
Flow Control	Experimental	1.74	0.358	138
	Control	1.55	0.431	174
Data Representation	Experimental	0.865	0.206	138
	Control	0.918	0.283	174
Abstraction	Experimental	0.813	0.397	138
	Control	0.913	0.398	174
User Interactivity	Experimental	1.22	0.326	138
	Control	1.35	0.290	174
Synchronization	Experimental	1.04	0.868	138
	Control	1.03	0.761	174
Parallelism	Experimental	0.747	0.520	138
	Control	0.640	0.472	174
Logic	Experimental	0.023	0.059	138
	Control	0.199	0.216	174

Considering the test of between-subject effects (see Table 3), the two-factor ANOVA showed a significant main effect on logic scores ($F(1, 310) = 0.002$, $p < 0.05$; partial eta-squared = 0.265; power = 0.899). As a result, it can be concluded that the original CT curriculum had a statistically significant impact on students' learning of logic as compared to the accessible CT curriculum.

Table 3. Test of between-subject effects (ANOVA).

Dependent Variable	Type III Sum of Squares	df	Mean Square	F	Sig.	Partial Eta-Squared	Observed Power
Flow Control	0.302	1	0.302	1.96	0.172	0.059	0.273
Data Representation	0.023	1	0.023	0.386	0.539	0.012	0.092
Abstraction	0.082	1	0.082	0.516	0.478	0.016	0.107
User Interactivity	0.138	1	0.138	1.43	0.240	0.044	0.213
Synchronization	0.000	1	0.000	0.000	0.987	0.000	0.050
Parallelism	0.095	1	0.095	0.383	0.541	0.012	0.092
Logic	0.256	1	0.256	11.2	0.002 *	0.265	0.899

Note: * $p < 0.05$.

In terms of the multivariate (MANOVA) test results (see Table 4), the original CT curriculum was found to cause a statistically significant difference in all CTC scores as a composite construct ($F(7, 304 = 0.028$, $p < 0.05$; Wilk's Lambda = 2.78, partial eta-squared = 0.438; observed power = 0.817). Therefore, it is safe to report that the original CT curriculum had a statistically significant impact on students' learning of CTCs as a single construct. Therefore, based on the project scores, students in the control group who learned with the original CT curriculum were more effective in applying their knowledge of the seven CTCs to their artifacts than students in the experimental group who learned with the accessible CT curriculum.

Table 4. Test of between-subject effect (MANOVA) on all seven CTC scores as a single construct.

Effect	Value	F	Hypothesis df	Error df	Sig.	Partial Eta-Squared	Observed Power
Wilks' Lambda	0.562	2.78	7.00	308	0.028	0.438	0.817

4.2. Correlation Analyses

Regardless of the group they were in (the experimental or control), students developed 312 computing artifacts in 24 sessions. The Pearson product moment correlation analyses between the seven CTCs (individually and collectively as a single construct)

and the 24 sessions were conducted to understand how the gradual change over the 24 sessions occurred.

Since CTCs, throughout the research project, were implemented in progressively increasing complexity and variety, students' learning of CTCs as exhibited on their projects was expected to show more in-depth and breadth coverage of the CTCs; that is, the scores on CTCs, individually and collectively, were expected to show a correlation to the progression of the sessions throughout the project year. This would reveal their progressive improvements in learning CTCs, individually and collectively.

Table 5 shows the descriptive statistics for the CTC scores of the projects developed by the students in both groups. The mean value (mean = 1.655) was the highest for flow control, while it was the lowest (mean = 0.103) for logic. This means that students across the two groups were able to apply their knowledge of flow controls (loops such as repeat, forever, etc.) better than any other CTCs. The descriptive statistics also show that students had difficulty learning logic (conditionals such as if, if-else, etc.).

Table 5. Descriptive statistics on correlations between computing projects and instructional sessions in both groups.

	Mean	Std. Deviation	N
Flow Control	1.655	0.399	20
Data Representation	0.888	0.241	20
Abstraction	0.858	0.394	20
User Interaction	1.284	0.312	20
Synchronization	1.034	0.808	20
Parallelism	0.699	0.494	20
Logic	0.103	0.174	20

Table 6 shows that there were statistically significant correlations between sessions and data representation (r = 0.456, p = 0.001), abstraction (r = 0.494, p = 0.001), synchronization (r = 0.650, p = 0.001), parallelism (r = 0.394, p = 0.05), and all CTCs as a combined construct (r = 0.754, p = 0.001). This indicates that both accessible and original CT curriculums were statistically significant in gradually improving the application of students' knowledge in data representation (data, variables, values, etc.), abstraction, synchronization (such as the use of wait, wait until, etc. blocks), and parallelism, as well as all CTCs combined.

Table 6. Correlations between CTCs and sessions across groups.

CTC Type	Pearson Correlation	Sig (2-Tailed)	N (Sessions)
Flow Control	0.043	0.811	24
Data Representation	0.456	0.008 **	24
Abstraction	0.494	0.003 **	24
User Interaction	−0.103	0.568	24
Synchronization	0.650	<0.001 **	24
Parallelism	0.394	0.023 *	24
Logic	0.185	0.301	24
All CTCs	0.754	<0.001 **	24

Note: * $p < 0.05$; ** $p < 0.01$.

Separate Pearson product moment correlations were calculated between sessions and CTCs, individually and combined, for each group to look at the gradual increase for each group. Descriptive statistics (see Table 7 below) for students in the experimental and control groups showed that the mean CTC value for both groups was the highest for flow control (experimental = 1.74; control = 1.55) and the lowest for logic (experimental = 0.02; control = 0.2). This means that both CT curriculums were the most and least successful in improving the application of students' knowledge of flow control and logic, respectively.

Table 7. Descriptive statistics for individual groups.

	Experimental Group			Control Group		
	Mean	**Std. Deviation**	**N**	**Mean**	**Std. Deviation**	**N**
Flow Control	1.74	0.36	18	1.55	0.43	15
Data Representation	0.86	0.21	18	0.92	0.28	15
Abstraction	0.81	0.40	18	0.91	0.40	15
User Interaction	1.22	0.33	18	1.35	0.29	15
Synchronization	1.04	0.87	18	1.03	0.76	15
Parallelism	0.75	0.52	18	0.64	0.47	15
Logic	0.02	0.06	18	0.2	0.22	15
Total	6.46	1.41	18	6.61	1.42	15

Based on the results of the Pearson product moment correlations (see Table 8 below), there were statistically significant correlations between sessions and data representation (r = 0.509, p = 0.05), sessions and synchronization (r = 0.611, p = 0.001), and sessions and all CTCs as a combined construct (r = 0.678, p = 0.001) for students in the experimental group. In comparison, there were statistically significant correlations between sessions and abstraction (r = 0.617, p = 0.05), synchronization (r = 0.720, p = 0.001), and all CTCs as a combined construct (r = 0.852, p = 0.001) for students in the control group.

Table 8. Correlations between CTCs and sessions for individual groups.

	Experimental Group			Control Group		
CTC Type	**Corr.**	**Sig (2-Tailed)**	**N**	**Corr.**	**Sig (2-Tailed)**	**N**
Flow Control	−0.133	0.600	19	0.293	0.289	15
Data Representation	0.509	0.031 *	19	0.408	0.131	15
Abstraction	0.385	0.114	19	0.617	0.014 *	15
User Interaction	−0.057	0.824	19	−0.237	0.396	15
Synchronization	0.611	0.007 **	19	0.720	0.002 **	15
Parallelism	0.420	0.083	19	0.398	0.141	15
Logic	−0.014	0.956	19	0.245	0.378	15
All CTCs	0.678	0.002 **	19	0.852	<0.001 **	15

Note: * $p < 0.05$; ** $p < 0.01$.

5. Discussion

This study aimed to determine the effectiveness of an accessible CT curriculum for students with ASD in improving their learning of CTCs (flow control, data representation, abstraction, user interactivity, synchronization, parallelism, and logic). Students' learning of CTCs was assessed via the scores of computing projects developed by them. The independent variable was the experimental group taught using the accessible CT curriculum, and the control group attending the session under the original CT curriculum.

Students in both groups were equal in their knowledge of CTCs at the pretest. Students developed a total of 312 (138 in the experimental and 174 in the control group) computing artifacts in this research project. The analysis of these artifacts indicated that the original CT curriculum had a statistically significant impact on students' learning of logic as compared to the accessible CT curriculum. Additionally, when all seven CTCs were examined, it was found that the original CT curriculum was statistically significantly more effective than the accessible CT curriculum in helping the students learn the seven CTCs as a single construct. This unexpected result may be attributed to the fact that the participating classroom teachers in the control group who implemented the original CT curriculum were not prevented from making their own modifications to the curriculum as they saw fit, based on the characteristics of their students. Some of the modifications applied by these teachers to the original CT curriculum included utilizing symbols, dividing the instructions into smaller sections, simplifying the instructional language, and eliminating some of the sessions due to their complexity. This is evident by the results that, regardless of the groups to which the students belonged, the research project made a statistically significant impact on students' learning of CTCs as a single construct and on their learning of data representation, abstraction, synchronization, and logic, individually.

When projects developed by students in both groups were correlated over the 24 sessions, statistically significant correlations were found between sessions and data representation, abstraction, synchronization, parallelism, and all CTCs as a single construct. This indicates that disregarding the group membership, the research study yielded statistically significant results highlighting a gradual improvement in students' learning of CTCs in data representation, abstraction, synchronization, parallelism, and all CTCs as a combined construct, as exhibited in their computing projects.

When the correlations were analyzed for each group, it was found that both the accessible and the original CT curriculum were statistically significantly effective in gradually increasing students' learning of CTCs in synchronization and all CTCs combined. Additionally, the accessible and original CT curriculums were more effective in improving students' learning of data representation and abstraction, respectively, as revealed in their computing projects.

These findings have several implications for computing curriculum design and development for students with ASD. First and foremost, students with ASD can effectively learn computing to contribute to the field of computing in every aspect on an accessible curriculum or on a curriculum designed for mainstream students as long as it is adjusted by a certified classroom teacher for students with ASD when being delivered. Secondly, the findings imply that more effective instructional methods must be explored for delivering flow control and logic, the two concepts that did not show significant improvement over the 24 sessions. Thirdly, this study shows that, for students with ASD, it is easier to learn flow controls (loops such as repeat, forever, etc.) and more difficult to learn logic (conditionals such as if, if-else, etc.) than any other CTCs on a CT curriculum that is adjusted based on their learning characteristics.

Specifically, the computing projects developed by the students had a mean score of 1.74 on flow controls, which corresponds to the 58th percentile in Dr. Scratch's [28] scoring system. Dr. Scratch [28] assigns each concept a score between 0 and 3 points. This score (1.74) indicates that students were able to control the behavior of characters by managing the flow of their programs. For instance, they utilized "repeat-until", "wait", "forever", "wait-until", and "repeat-until" blocks to control program flow and character behavior. For example, they made the Scratch cat character jump a certain number of times using loops.

On the other hand, the projects developed by the students revealed a score of 0.23 on logic, indicating difficulty integrating logic-related blocks into their computing projects. In other words, they struggled with using even the most basic logic block, such as "if", in their projects. "if-else" and multiple "if" or "if-else" blocks were not utilized at all. Consequently, students were unable to program their characters to behave differently based on certain conditions, such as making the character bounce if it hits the boundaries of the stage, or utilizing multiple conditions to have their characters behave differently based on the respective condition, such as bouncing back if it touches the boundaries of the stage or disappearing if it touches another character on the stage.

A future study may be conducted to see if an accessible computing curriculum would be effective in grade levels other than seventh grade, which was the subject of this study. Involving a larger number of students may lead to different results in terms of students' learning of CTCs, which also requires further exploration. Finally, in a future study, it is worth finding if there is a statistically significant correlation between sessions and an increase in students' project scores, indicating their development of fluency in the CTPs affected by the curriculum.

Limitations in this study included the nature of students with ASD, particularly the fact that each student with ASD is unique. The study comprised all seventh-grade students who volunteered to participate without any exclusion criteria, and those who continued to participate were permitted to stay in the project. The data generated by these students were included in the study, which might have influenced the results. Furthermore, the experimental and control groups originated from two different schools with different teachers. The implementation of the computing curriculums varied in these two schools due

to differences in teachers, administration, staffing, and available equipment and facilities. Therefore, these discrepancies may have influenced the results.

Author Contributions: Conceptualization, A.A. and M.L.B.; methodology, A.A., M.L.B. and G.V.B.; validation, A.A., M.L.B., G.V.B. and K.P.; formal analysis, A.A., M.L.B., G.V.B., K.P. and R.I.; investigation, A.A., M.L.B., G.V.B., K.P. and R.I.; resources, A.A., M.L.B., G.V.B., K.P. and R.I.; writing—original draft preparation, A.A.; writing—review and editing, A.A. and F.Y.; project administration, A.A., M.L.B. and F.Y.; funding acquisition, A.A. and M.L.B. All authors have read and agreed to the published version of the manuscript.

Funding: This material is based upon work supported by the National Science Foundation under Grant No. 2031427. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the National Science Foundation.

Institutional Review Board Statement: This study was conducted in accordance with the Declaration of Helsinki and approved by the Institutional Review Board (or Ethics Committee) of Youngstown State University (protocol code #005-21 and date of approval 21 October 2020).

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study and their parents/caregivers.

Data Availability Statement: The data presented in this study are available upon request from the corresponding author. The data are not publicly available due to the minority status and special characteristics of participants.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Brackmann, C.P.; Román-González, M.; Robles, G.; Moreno-León, J.; Casali, A.; Barone, D. Development of Computational Thinking Skills through Unplugged Activities in Primary School. In Proceedings of the 12th Workshop on Primary and Secondary Computing Education, WiPSCE '17, New York, NY, USA, 8–10 November 2017; pp. 65–72. [CrossRef]
2. Folk, R.; Lee, G.; Michalenko, A.; Peel, A.; Pontelli, E. GK-12 DISSECT: Incorporating Computational Thinking with K-12 Science without Computer Access. In Proceedings of the Frontiers in Education Conference, El Paso, TX, USA, 21–24 October 2015; Volume 2014, pp. 1–8. [CrossRef]
3. Goldberg, D.S.; Grunwald, D.; Lewis, C.; Feld, J.A.; Hug, S. Engaging Computer Science in Traditional Education: The ECSITE Project. In Proceedings of the ACM Conference on Innovation and Technology in Computer Science Education, Haifa, Israel, 3–5 July 2012; pp. 351–356. [CrossRef]
4. Hambrusch, S.; Hoffmann, C.; Korb, J.T.; Haugan, M.; Hosking, A.l. A Multidisciplinary Approach towards Computational Thinking for Science Majors. *SIGCSE Bull. Inroads* **2009**, *41*, 183–187. [CrossRef]
5. Burgett, T.; Folk, R.; Fulton, J.; Peel, A.; Pontelli, E.; Szczepanski, V. DISSECT: Analysis of Pedagogical Techniques to Integrate Computational Thinking into K-12 Curricula. In Proceedings of the Frontiers in Education Conference, El Paso, TX, USA, 21–24 October 2015; Volume 2014, pp. 1–9. [CrossRef]
6. Qin, H. Teaching Computational Thinking through Bioinformatics to Biology Students. *SIGCSE Bull. Inroads* **2009**, *41*, 188–191. [CrossRef]
7. Jenkins, J.T.; Jerkins, J.A.; Stenger, C.l. A Plan for Immediate Immersion of Computational Thinking into the High School Math Classroom through a Partnership with the Alabama Math, Science, and Technology Initiative. In Proceedings of the Annual Southeast Conference, Tuscaloosa, AL, USA, 29–31 March 2012; pp. 148–152. [CrossRef]
8. Miller, R.B.; Kelly, G.N.; Kelly, J.T. Effects of Logo Computer Programming Experience on Problem Solving and Spatial Relations Ability. *Contemp. Educ. Psychol.* **1988**, *13*, 348–357. [CrossRef]
9. Grover, S.; Cooper, S.; Pea, R. Assessing Computational Learning in K-12. In Proceedings of the 2014 Innovation and Technology in Computer Science Education Conference, Uppsala, Sweden, 21–25 June 2014; pp. 57–62. [CrossRef]
10. Haddad, R.; Kalaani, Y. Can Computational Thinking Predict Academic Performance? In Proceedings of the 5th IEEE Integrated STEM Education Conference, Princeton, NJ, USA, 7 March 2015; pp. 225–229. [CrossRef]
11. De Oliveira, O.L.; Del Val Cura, L.M.; Do Carmo Nicoletti, M. Quantitative Correlation between Ability to Compute and Student Performance in a Primary School. In Proceedings of the 45th ACM Technical Symposium on Computer Science Education, Atlanta, GA, USA, 5–8 March 2014; pp. 505–510. [CrossRef]
12. Kannadass, P.; Hidayat, R.; Siregar, P.S.; Husain, A.P. Relationship Between Computational and Critical Thinking Towards Modelling Competency Among Pre-Service Mathematics Teachers. *TEM J.* **2023**, *12*, 1370–1382. [CrossRef]
13. Peng, H.-H.; Murti, A.T.; Silitonga, L.M.; Wu, T.-T. Effects of the Fundamental Concepts of Computational Thinking on Students' Anxiety and Motivation toward K-12 English Writing. *Sustainability* **2023**, *15*, 5855. [CrossRef]

14. Prince, T. Enhancing Linear Algebra Learning through Computational Thinking: A Project-Based Approach. *HETS Online J.* **2023**, *13*, 132. [CrossRef]
15. Robledo-Castro, C.; Castillo-Ossa, L.F.; Hederich-Martínez, C. Effects of a Computational Thinking Intervention Program on Executive Functions in Children Aged 10 to 11. *Int. J. Child Comput. Interact.* **2023**, *35*, 100563. [CrossRef]
16. Güven, I.; Gulbahar, Y. Integrating Computational Thinking into Social Studies. *Soc. Stud.* **2020**, *111*, 234–248. [CrossRef]
17. Hava, K.; Koyunlu Ünlü, Z. Investigation of the Relationship Between Middle School Students' Computational Thinking Skills and Their STEM Career Interest and Attitudes Toward Inquiry. *J. Sci. Educ. Technol.* **2021**, *30*, 484–495. [CrossRef]
18. Wing, J. Computational Thinking. *Commun. ACM* **2006**, *49*, 33–35. [CrossRef]
19. Mumcu, F.; Kıdıman, E.; Özdinç, F. Integrating Computational Thinking into Mathematics Education through an Unplugged Computer Science Activity. *J. Pedagog. Res.* **2023**, *7*, 72–92. [CrossRef]
20. Waris, Z.; Bibi, H. Advancing Students' Computational Thinking Skills: A Study on Elementary Level. *Pak. J. Soc. Sci.* **2023**, *43*, 661–672. [CrossRef]
21. Wille, S.; Century, J.; Pike, M. Exploratory Research to Expand Opportunities in Computer Science for Students with Learning Differences. *Comput. Sci. Eng.* **2017**, *19*, 40–50. [CrossRef]
22. Arslanyilmaz, A.; Briley, M.; Loto, M.B.; Fernberg, C.; Beadle, G.; Coldren, J. An Accessible Computing Curriculum for Students with Autism Spectrum Disorders. In Proceedings of the Society for Information Technology & Teacher Education International Conference, Online, 29 March 2021; Langran, E., Archambault, L., Eds.; Association for the Advancement of Computing in Education: Waynesville, NC, USA, 2021; pp. 17–23.
23. Brennan, K.; Balch, C.; Chung, M. Creative Computing: Scratch Curriculum Guide. 2014. Available online: https://scratched.gse.harvard.edu/guide (accessed on 1 September 2023).
24. Brennan, K.; Resnick, M. New Frameworks for Studying and Assessing the Development of Computational Thinking. In Proceedings of the 2012 Annual Meeting of the American Educational Research, Vancouver, BC, Canada, 13–17 April 2012.
25. Scratch—Imagine, Program, Share. Available online: https://scratch.mit.edu/ (accessed on 5 February 2024).
26. ISAC NSF—YouTube. Available online: https://www.youtube.com/ (accessed on 6 February 2024).
27. Boe, B.; Hill, C.; Len, M.; Dreschler, G.; Conrad, P.; Franklin, D. Hairball: Lint-Inspired Static Analysis of Scratch Projects. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education (SIGCSE '13)*; ACM: Denver, CO, USA, 2013; pp. 215–220.
28. Dr. Scratch. Available online: http://www.drscratch.org/ (accessed on 5 February 2024).