

## PAPER

# Making Computer Science Accessible through Universal Design for Learning in Inclusive Education

Gulnaz Salgarayeva, Aigul  
Makhanova(✉)

Kazakh National Women's  
Teacher Training University,  
Almaty, Kazakhstan

[makhanova.a@qyzpu.edu.kz](mailto:makhanova.a@qyzpu.edu.kz)

## ABSTRACT

The field of technology and computer science (CS) is developing dynamically. Just as anyone can learn computers at any age, students with special educational needs (SEN) also aspire to acquire IT (information technology) knowledge on an equal footing with all other students. However, one of the obstacles facing students with SEN is the lack of educational materials and programs for CS in secondary schools. The authors have designed teaching materials and assignments that promote inclusion. This study aims to evaluate the impact of teaching resources developed based on universal design for learning (UDL) to make the school's CS course accessible to all students. The experiment involved 16 students and five teachers. For 8 weeks, students studied computer science using training materials based on UDL. Assessment of knowledge outcome indicators, particularly programming skills, was conducted before and after the experiment. After studying computer science through specific tasks, the interviewees demonstrated a higher level of assimilation of the subject, as indicated by the subsequent test results (mean = 12.13, standard deviation = 1.20), compared to the pre-experimental test (mean = 8.94, standard deviation = 1.12). The study demonstrated that using special UDL-based tasks to teach CS makes it more accessible and has a positive impact on students with special educational needs.

## KEYWORDS

computer science (CS), equity, inclusive education, instructional materials, teaching strategies, universal design for learning (UDL)

## 1 INTRODUCTION

An accessible computing curriculum is a way to enhance the relevance of research on human-computer interaction. It ensures that computer science (CS) teaching materials are designed based on universal learning principles for the benefit of all students, including those with special educational needs (SEN) [1].

A popular approach to sparking students' interest in CS is to engage secondary school students in classes that teach programming in an exciting manner.

Salgarayeva, G., Makhanova, A. (2024). Making Computer Science Accessible through Universal Design for Learning in Inclusive Education. *International Journal of Engineering Pedagogy (IJEP)*, 14(5), pp. 109–122. <https://doi.org/10.3991/ijep.v14i5.48811>

Article submitted 2024-03-04. Revision uploaded 2024-04-07. Final acceptance 2024-04-07.

© 2024 by the authors of this article. Published under CC-BY.

Hansen describes how a group of CS teachers should integrate programming as one of the methods for teaching engineering design to students [2]. Given that a significant amount of emphasis is placed on teaching the programming aspect of the CS course outside regular school hours, whether as an elective or an additional course [3], it is crucial that CS is thoroughly integrated into the curriculum to guarantee accessibility for all students, including those with special educational needs.

Universal Design for Learning (UDL) is a set of principles that enable teachers to develop instruction to meet the diverse needs of all students. By using UDL in the development of teaching materials used during lessons, computer instruction can also be adapted for children with special educational needs [4].

According to the National Center on UDL [5], “UDL provides a blueprint for creating instructional goals, methods, materials, and assessments that work for everyone. It is not a one-size-fits-all solution but rather flexible approaches that can be customized and adjusted to individual needs.”

Universal Design for Learning, which is based on three basic principles, reduces barriers to learning by making content accessible to all students, actively involving them in practice, and helping them demonstrate the knowledge they have acquired. For example, a teacher can present information to a student in a variety of formats: paper, electronic, audio, or video. Similarly, the answer can be obtained in several ways: written, oral, in the form of images, or through a computer program.

The aim of this study is to investigate how teaching materials grounded in UDL impact the holistic development of CS in students with special educational needs.

Our study addresses the following two research questions (RQ):

**RQ1:** How does the use of tasks based on UDL in CS teaching impact students with special educational needs?

**RQ2:** What difficulties do students with special educational needs encounter when completing tasks based on UDL?

The remainder of this paper is organized as follows: Section 2 presents a comprehensive review of the literature on the research topic discussed here. The methodological aspects are detailed in Section 3. Section 4 describes the main results obtained. These results are discussed in Section 5. Section 6 concludes by presenting the main findings of this study and outlining future research prospects.

## 2 RELATED WORK

### 2.1 User-friendly programming environments and educational programs

Digital technologies, especially specialized training programs and mobile applications, can serve as auxiliary tools in various living conditions [6]. Currently, the popularity of programming environments and user-friendly educational programs continues to grow. In addition, there are active projects aimed at attracting students with special educational needs to study in the field of computer science.

In the work by R. Ladner and A. Stefik [7], the activities of the Access for All project concerning the development of adapted learning tools and curricula and the professional development of CS teachers are discussed.

Schanzer [8] described courses on effective teaching of CS on the largest educational platform, Bootstrap, for all students. Leigh Ann DeLyser [9] analyzed research

and best practices related to supporting CS education. Researchers examined the CSForAll platform, which develops educational programs for teaching CS and works on the professional development of teachers in collaboration with schools that implement inclusive education.

The study by H. Bučková and J. Dostál [10] focused on the inclusive features of the Code.org platform for every student in the school.

Teachers can use these tools for self-study or to increase their students' interest in computer science.

## 2.2 Universal Design for Learning as a method of promoting inclusive education

Universal Design for Learning, which enables teachers to create learning materials for all students, including children with special educational needs, is one of the methods for advancing the practice of inclusive education.

The results of the analysis conducted in the scientific study by M.W. Ok et al. [11] demonstrated that incorporating teaching materials in UDL-based courses can have a positive impact on the full participation of students with special educational needs in the general education curriculum. It can also enhance the accessibility of subject content, improve students' academic performance, and foster social interaction.

In their work, A.B. Mourão and J.F. Netto [12] provide a practical example of using an inclusive model for developing and evaluating accessible and adapted CS teaching materials. This work has helped to stimulate the promotion of the importance of social and digital integration in the educational process for students with special educational needs.

The results of the study by Macedo et al. [13] have made a significant contribution to the development of accessible learning materials. This work provides information that teachers can use to guide and support the development of digital learning materials with accessible features and functions.

The study by H. Abdellaoui et al. [14] highlights the importance of developing technological solutions that promote inclusive education. Ensuring the accessibility of educational content by adapting to the diverse needs of students with SEN helps create accessible learning systems.

Researchers J. Nganji and M. Brayshaw [15] are investigating the customization of learning materials for students with special educational needs. In addition, the authors express confidence that in the future, developers will create fully inclusive virtual learning environments. They recognize the current limitations of instructional materials in terms of universality while striving to ensure accessibility for students with special educational needs.

C. Shelton [16], in his research, examines the literature on inclusive education in computer courses with a focus on establishing an inclusive framework. The researcher identifies a number of inclusive practices, including combating prejudice. The results of the study emphasize the necessity for additional research in creating an inclusive computer classroom.

R. Ladner and M. Israel [17] have identified three main aspects that children with special educational needs overlook when studying CS: teachers' attitudes towards students, the teaching methods employed, and the accessibility of resources. The scientists cited note that if a teacher reduces learning outcome requirements because

of the student's special needs, this may unconsciously provoke a negative attitude towards the student's inability to master computers.

Universal Design for Learning incorporates features that are advantageous for some students, helpful for others, and not detrimental to anyone, guaranteeing that all students, including those requiring special education, attain high outcomes [18].

### 2.3 Development of algorithmic thinking in children with UDL based on the Use-Modify-Create approach

Differences in students' levels of algorithmic thinking can manifest in any computer class, potentially associated with various aspects of inclusion.

O. Hatlevik and K. Christophersen [19] demonstrate that some differences in students' digital competence are linked to their cultural background and mother tongue. The studies reviewed have not fully addressed the question of how teachers can effectively adapt to the wide range of learning outcomes expected in the classroom, particularly in computer courses [20]. It is clear that inclusive pedagogical approaches, such as use, modify, and create (UMC), successfully adapted to a variety of disciplines, can also be adopted for CS teaching [21].

In their qualitative study, M. Israel et al. [22] examined how teachers used UDL to teach CS to students with varying levels of competence. The emphasis was placed on breaking down tasks into manageable parts, allowing students to make choices, and providing multiple means of response. In their work, the authors propose projects based on the "Use-Modify-Create" approach as a practical example of applying UDL. Students start by using the code of a ready-to-use program, modifying it, and attempting to transform and develop it themselves. The authors believe that this method enables students to become fully immersed in the educational process and work independently.

M. Israel et al. [23] found that finding ways to make computing accessible and interesting to a wide range of students is very demanding and revealed that the resources available to help teachers in this direction are scarce.

The research by N. Lytle et al. [24] suggests that using UDL with the Use-Modify-Create (UMC) approach has an impact on the training and development of students' algorithmic thinking skills, as well as on the easy assimilation of tasks considered "too difficult" for some students.

I. Lee et al. [25] have promoted the UMC approach not only as a basis for completing the lesson based on UDL but also as a means of creating the conditions for the full participation of all students in the educational process of CS teaching.

The results of the literature review indicate that although pedagogical approaches have shown potential in assisting schools and teachers in establishing more inclusive classrooms, additional research is required to guarantee inclusivity in CS education.

## 3 MATERIALS AND METHODS

The methodology of this study can be summarized as follows:

- Conduct a pre-test to determine students' level of learning in information science;
- Utilize UDL tasks based on the UMC approach for information science learning during the study;

- Obtain a post-test to identify the impact of UDL tasks on students' comprehensive development in information science and compare the results;
- Implement the practice of using UDL-based tasks to enhance students' programming skills.

### 3.1 User-friendly programming environments and educational programs

This study was conducted in inclusive classes of a general secondary school with the participation of 16 students aged 12 to 15 years (7 students from grade 6, 9 students from grade 7) and 5 CS teachers. The teaching process of CS has been monitored for 8 weeks.

Before the experiment, the subject of CS was taught using the traditional method, where teachers were the primary source of knowledge for teaching students. They assigned tasks to all students that were not based on a universal design for learning.

At the pre-test, students received an assessment to evaluate their syntax skills (4 questions), computational thinking skills (4 questions), creative skills (4 questions), and interdisciplinary skills (4 questions), totaling 16 questions. In compiling this test, we were guided by the concept of the Four Layers of Programming Skills proposed by K. McGillivray [26].

During the experiment, CS assignments based on UDL were used. Further details are provided in the following subsection of the paper. At the end of the experiment, a post-test was administered using the same test as the pre-test to assess the impact on the overall development of CS skills among students with special educational needs and to compare the results.

To analyze the data obtained in the study, we utilized statistical hypothesis testing. As the measurements were carried out with the same group of students at regular intervals before and after the experiment, we used the student's t-test to calculate the empirical value of the t-test when testing the difference between two dependent paired samples.

### 3.2 Respondent data

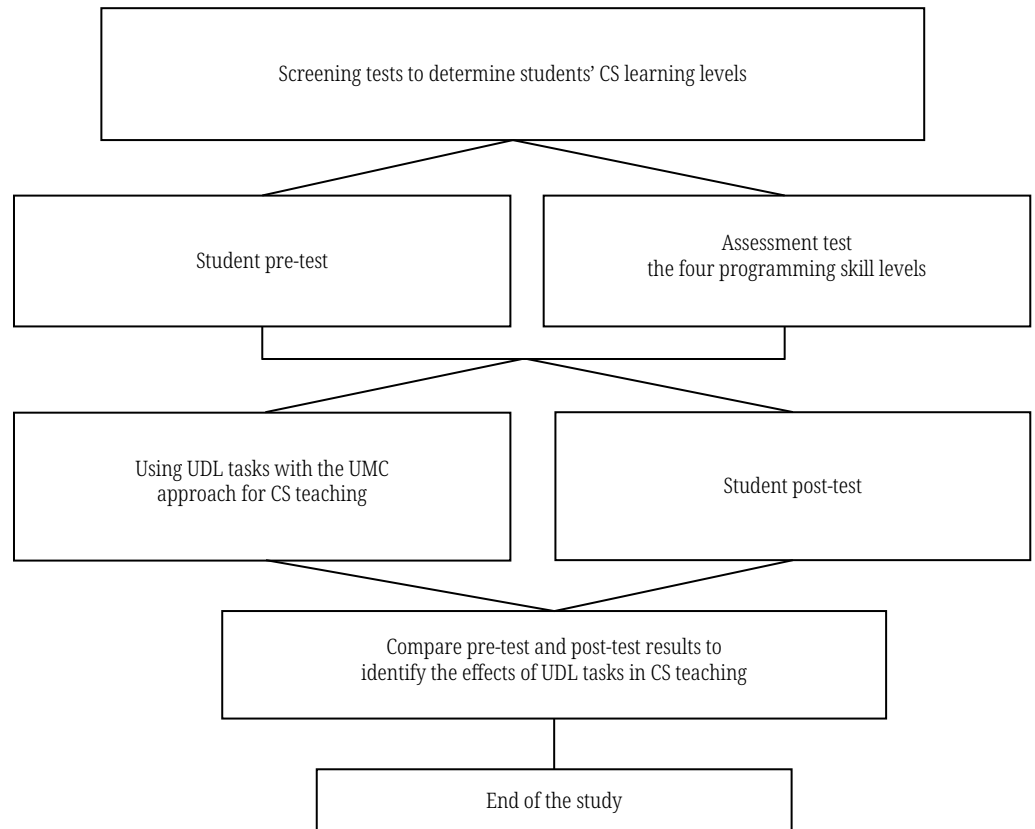
The respondent information is coded (refer to Table 1). Participating students study according to an individualized education program in inclusive classes at general secondary schools. Teachers are involved in the experiment at the organizational level and teach social sciences in these classes. During the experiment, CS assignments based on a UDL were used for the students, and the teachers attempted to identify the challenges faced by the students while completing UDL tasks.

### 3.3 Experimental procedure

The procedure of the study is illustrated in Figure 1. The selection procedure for participants began with obtaining permission to conduct research at a general secondary school with inclusive classes. The consent of the parents of students with special educational needs participating in the study was obtained. CS teachers voluntarily participated in the study.

**Table 1.** Participants data

Code	Gender	Age	Grade/Status	Diagnosis
P1	Male	12	6th grade	Developmental language disorder
P2	Male	12	6th grade	Developmental language disorder
P3	Male	13	6th grade	Cognitive dysfunction
P4	Female	12	6th grade	Cognitive dysfunction
P5	Female	13	6th grade	Cognitive dysfunction
P6	Male	13	6th grade	Cognitive dysfunction
P7	Female	14	6th grade	Phonetic-phonematic underdevelopment of speech
P8	Male	14	7th grade	Phonetic-phonematic underdevelopment of speech
P9	Female	14	7th grade	Cognitive dysfunction
P10	Female	14	7th grade	Developmental language disorder
P11	Male	14	7th grade	Cognitive dysfunction
P12	Male	15	7th grade	Cognitive dysfunction
P13	Male	14	7th grade	Developmental language disorder
P14	Female	15	7th grade	Cognitive dysfunction
P15	Male	14	7th grade	Cognitive dysfunction
P16	Male	14	7th grade	Developmental language disorder



**Fig. 1.** Study process

### 3.4 Universal Design for Learning task description

In teaching CS, our goal was to enhance students' syntactic skills, programming thinking skills, creativity, and interdisciplinary skills through tasks grounded in universal design for learning.

In CS courses, the emphasis is on developing syntactic skills in the initial stages of programming instruction. Syntactic skills involve reading and writing a programming language by following the rules that dictate the use of various symbols.

The importance of developing programmatic thinking skills lies in the ability to think like a computer and translate the process into a computer-friendly set of instructions. In addition to intuitive and visual thinking, students are also capable of logical and verbal reasoning. In the implementation of the logical process, programmatic thinking is important. At a certain stage of programming implementation, creative skills will be required to write original code.

Creative skills are essential in all fields, not just programming. Writing code in programming can always be combined with creating a website on various topics or developing an application. Having knowledge in various fields, in addition to programming, will greatly assist in creating new projects.

The development of students' interdisciplinary skills will enhance their programming abilities by integrating their existing knowledge with their interests.

In order to develop the above-mentioned skills in students with SEN, training tasks based on the universal design for training were compiled using the UMC approach proposed by Israel et al. (2020).

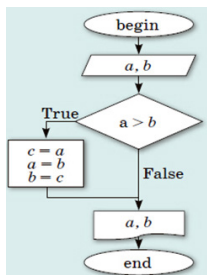
According to these tasks, in the first stage, "Use," students start by reproducing and reworking the code. In the second stage, "Modify," students try to debug the program when it does not work and restore the disassembled code. In the final stage, "Create," students attempt to expand the program to perform a task of their choice.

All versions of the assignment are available simultaneously, allowing students to switch between them as they learn. This helps individuals work more independently, reduces cognitive load, and provides comprehensive support. This approach illustrates that by developing activities that support students with special educational needs, we can enhance the understanding and proficiency of all our students.

In secondary schools, the subject of CS is studied according to the standard curriculum with updated content. In the CS course for grades 6–7 of basic secondary education, the topic "Programming in Python" is taught. In the sixth grade, students study the following topics: the alphabet of the language, syntax, data types, rules for writing arithmetic operations, input and output of numbers, and programming linear algorithms. In the seventh grade, students will explore topics such as working with files, programming branched algorithms, implementing nested and complex conditions, and organizing choices.

Since CS is taught on common topics in all regions of the country, educational materials were compiled to develop the programming skills of students with special educational needs. These materials are based on the UDL (refer to Table 2) and were derived from CS textbooks by G. Salgarayeva et al. [27–28], R. Kadirkulov et al. [29], and S. Mukhambetzhanova et al. [30].

**Table 2.** Examples of UDL tasks with UMC approach

Task	Version 1 Use	Version 2 Modify	Version 3 Create
Calculate the expression <b>12-6*2+9</b> in Python <i>(Task from 6 grade CS textbook by G. Salgarayeva et al., 2020)</i>	Use the operator <b>print()</b> to calculate the expression in Python.  <code>print(12-6*2+9)</code>	Enter the given 4 numbers using the <b>input()</b> operator and calculate the expression in Python.	Create a complex expression with <b>+, -, *, /</b> enter the numbers using the <b>input()</b> operator, and output the result using the <b>print()</b> operator.
Create a program to write the word <b>“Computer Science”</b> back in the form of <b>“ecneicS retupmoC”</b> . <i>(Task from 6 grade CS textbook by G. Salgarayeva et al., 2020)</i>	Use the <b>reverse line reading command</b> to complete the task.  <code>str="Computer Science" print (str[::-1])</code>	Write a program that reads both lines backwards, separating the words <b>“Computer”</b> and <b>“Science”</b> from the given word <b>“Computer Science”</b> .	Find a 4-line poem about <b>“Spring”</b> and create a program to write each line backwards.
If the first student collects <b>x</b> buckets of apples in 1 hour, the second student collects <b>y</b> buckets of apples, the third student collects <b>z</b> buckets of apples, how many buckets of apples do they collect in <b>t</b> hours? Create a program. <i>(Task from 6 grade CS textbook by G. Salgarayeva et al., 2020)</i>	Use the given algorithm to solve the task and check it in Python program.  <b>1. Start</b> <b>2. Input x</b> <b>3. Input y</b> <b>4. Input z</b> <b>5. Input t</b> <b>6. Calculate S=(x+y+z) *t</b> <b>7. Print S</b> <b>8. Stop</b>	If <b>t</b> hours and <b>S</b> are known as the total number of buckets of apples collected, then create a program to determine how many buckets of apples students collected in <b>1 hour</b> .	Create a task text for selling apples to grocery stores and develop a program that will determine the <b>expenses</b> and <b>profits</b> of the grower.
Given the variables a and b. Arrange their values in ascending order. <i>(Task from 7 grade CS textbook by G. Salgarayeva et al., 2021)</i>	<b>A flowchart</b> of the task is given. Write and test program code in Python.  	The order of the task program codes is set <b>incorrectly</b> . Place them in the correct order and check them in Python.  <code>if a&gt;b: a=b c=a b=c b=int(input()) a=int(input()) print(a,b)</code>	Given the variables a, b, c. Build a flowchart and a program that places their values in <b>descending order</b> .
After executing the program fragment, determine the value of the variable <b>a</b> .  <code>a=7 b=-10 if a&gt;1 or a&lt;b: a=5 elif a&gt;1 and a==b: a-=5 print(a)</code>  <i>(Task from 7 grade CS textbook by S. Mukhambetzanova et al., 2021)</i>	Check the given program fragment in the Python program. Explain the answer to the teacher <b>orally, visually</b> or in <b>writing</b> .	Change the initial value of <b>a</b> so that it is less than the value of <b>b</b> , and as a result, determine the value of the variable <b>b</b> .	Change the <b>conditions</b> of the given task and explain how the resulting <b>a</b> or <b>b</b> values are obtained.
Given the number <b>P</b> . If the number P is greater than 0, then find the cube of the number, if less than 0, then add the number <b>K</b> to it, if equal to 0, then subtract <b>Z</b> from the number. <i>(Task from 7 grade CS textbook by R. Kadirkulov et al., 2021)</i>	Use the given algorithm to solve the task and check it in Python program.  <b>1. Start</b> <b>2. Enter P</b> <b>3. Enter K</b> <b>4. Enter Z</b> <b>5. if P&gt;0 then P=P*P*P</b> <b>6. else if P&lt;0 then P=P+K else P=P-Z</b> <b>7. Print P</b> <b>8. Stop</b>	To check all <b>3 conditions</b> given in the content of the task, change the value of <b>P</b> and <b>orally</b> explain the results to the teacher.	Create other tasks by changing the <b>conditions</b> and value of <b>P, K, Z</b> . Explain to the teacher the value obtained as a result of the execution of the program.



These guidelines assist CS teachers in anticipating potential obstacles, planning activities to overcome them, or adapting existing teaching materials to engage students with special educational needs in the classroom as much as possible.

#### 4 MAIN RESULTS

Before pedagogical intervention, students with special educational needs were assessed for their programming skills level. After that, for eight weeks, they were taught CS classes with educational tasks based on UDL. The students were tested again to evaluate their programming skills after conducting the experiment.

Table 3 presents pretest and posttest results along with their corresponding descriptive statistics: N (number of respondents), SD (standard deviation), SE (standard error), and t-test.

**Table 3.** General results of teaching CS with UDL tasks

Test	N	Mean (M)	SD	SE	t	Critical t-Value
Pretest	16	8.94	1.12	0.28	7.74	2.131
Posttest	16	12.13	1.20	0.30		

Note: N-number, SD: Standard deviation, SE: Standard error.

The null hypothesis H0 posits that the improvement in programming skills among students learning CS through UDL-based tasks is affected by random factors rather than pedagogical factors. The alternative hypothesis (H1) posits that the pedagogical impact of utilizing UDL-based tasks influences the development of programming skills when teaching information science to students with special educational needs.

Based on the results, it is evident that implementing tasks based on UDL positively affects students with special educational needs when learning CS. The post-test result (M = 12.13, SD = 1.20) significantly exceeded the pre-test scores (M = 8.94, SD = 1.12). A significant difference is evident between the t-statistics (7.74) and the critical t-value (2.131). Consequently, the null hypothesis H0 is rejected, and the alternative hypothesis that the pedagogical influence introduced using tasks based on UDL had an impact on the development of programming skills in students with special educational needs to achieve positive learning outcomes in teaching CS is accepted.

The results of the study on the development of programming skills among students with special educational needs demonstrated a positive difference between the stages before and after the experiment. If we analyze each of the criteria for syntax skills, programmatic thinking skills, creative skills, and interdisciplinary skills separately, we can observe dynamics for each skill (refer to Table 4).

**Table 4.** Pre-testing and post-testing data of the development of programming skills

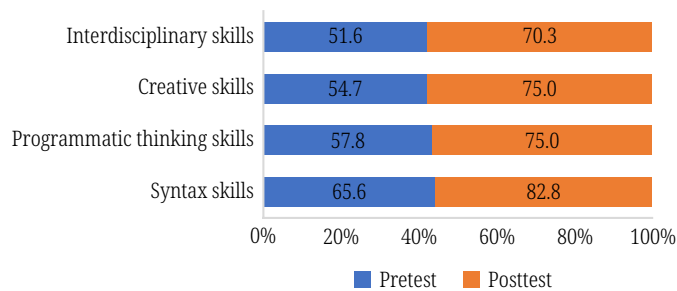
Skills	Test	Mean	SD	SE	t-Student
Syntax skills	Pretest	2.5625	0.51235	0.12809	4.23
	Posttest	3.3125	0.47871	0.11968	
Programmatic thinking skills	Pretest	2.3125	0.79320	0.19830	2.91
	Posttest	3.0000	0.51640	0.12910	

(Continued)

**Table 4.** Pre-testing and post-testing data of the development of programming skills (*Continued*)

Skills	Test	Mean	SD	SE	t-Student
Creative skills	Pretest	2.1875	0.91059	0.22765	2.78
	Posttest	3.0000	0.73030	0.18257	
Interdisciplinary skills	Posttest	1.8750	0.61914	0.15478	4.54
	Pretest	2.8125	0.54391	0.13598	

Figure 2 shows the percentage ratio of programming skill development in students with SEN. Prior to engaging in classes with UDL tasks, participants demonstrated a programming ability development of 65.8% in syntax skills. After the training, it increased to 82.8%. As for programmatic thinking skills, the percentage was 57.8% before the experiment, rising to 75.0% post-training. Before the experiment, students demonstrated a 54.7% improvement in creative skills, which increased to 75.0% after implementing UDL approaches. The indicator of interdisciplinary skills also increased, from 51.6% to 70.3%. In general, there is a positive difference in the development of programming skills among students with SEN before and after the experiment.

**Fig. 2.** Programming skills growth chart for students with SEN

To identify the challenges that students with special educational needs face when completing educational tasks based on UDL, we conducted interviews with CS teachers. Teachers noted that completing Version 1 tasks did not pose difficulties for students. The students wrote a program based on a given algorithm or checked the program code. When completing Version 2 tasks, students needed to arrange the correct sequence of steps to achieve their goal. Students needed advice from a teacher to complete the tasks. All five teachers noted that they observed students who could not receive feedback in time quickly shifting their focus from one task to another. When completing Version 3 tasks, there were no restrictions on students' creativity. The students were given the opportunity to choose and exercise independently at their discretion.

## 5 DISCUSSION

In our study, we considered two research questions. According to the first research question, we investigated the impact of educational tasks based on UDL on teaching CS to students with special educational needs. We used the "UMC" approach when developing educational tasks based on UDL. This study contributes

to the development of inclusive practices in teaching CS to students with special educational needs.

A research team at the Creative Technology Research Lab (CTRL) in the USA [22] studied how teachers implement UDL in CS education. Israel et al. [22] collaborated with four CS teachers in U.S. elementary schools to teach and observe the implementation of UDL. The study revealed that the teachers introduced several universal learning techniques into the learning process. For example, they allowed their students to choose their own task types and presented material in various formats (oral, written, and visual). Despite the small sample size, the researchers noted the flexibility of the UDL approach to meet diverse needs in different learning situations. Further research on a broader range of task models will be necessary to assess the effectiveness of this approach.

To find the answer to the second research question, we relied on the responses of teachers who organized computer courses during the study. The aim of the second research question was to identify challenges in completing tasks based on a universal understanding of learning. Teachers pointed out that the main difficulties were concentrating on something else when completing a task, needing the teacher's usual help, and expecting support from the teacher. At the end of the experimental period, the teachers observed that task completion by uninstructed students without teacher intervention resulted in an increase in their self-confidence.

## 6 CONCLUSIONS

The results of this research work offer a different perspective on the possibilities of applying UDL tasks in inclusive education. The use of adapted tasks enhances opportunities for full participation in the educational process for students with special educational needs in CS education.

The results of a study conducted with students from mainstream secondary schools engaged in inclusive education using tasks based on UDL showed an involved impact on student outcomes. Results obtained after the experiment showed that students' programming skills had improved significantly. In this respect, we are convinced that CS teachers will pay attention to the widespread use of tasks based on UDL and include inclusive classes in secondary schools in the teaching system.

This study is part of an approach aimed at making knowledge accessible to all children through a universal design for learning, akin to what some authors have described as sustainable pedagogy [31]. In this way, we ensure the full participation of all children in the learning process, including those with special educational needs.

To enhance the implementation of UDL-based tasks in inclusive education and achieve more comprehensive results, the authors propose the following additions to the current study:

- Continued research into the use of tasks based on UDL to ensure the accessibility of CS education.
- Further research should be conducted on the use of UDL-based tasks to achieve even higher indicators of students' algorithmic thinking.
- Enhance the implementation of tasks based on UDL to ensure that knowledge is accessible to all students across various subjects.

## 7 REFERENCES

- [1] C. Stephanidis, G. Salvendy, M. Antona, J. Y. C. Chen, J. Dong, V. G. Duffy, X. Fang, C. Fidopiastis, G. Fragomeni, L. P. Fu, Y. Guo, D. Harris, A. Ioannou, K. Jeong, S. Konomi, H. Krömker, M. Kurosu, J. R. Lewis, A. Marcus, ... J. Zhou, "Seven HCI grand challenges," *International Journal of Human Computer Interaction*, vol. 35, no. 14, pp. 1229–1269, 2019. <https://doi.org/10.1080/10447318.2019.1619259>
- [2] A. Hansen, A. Iveland, H. Dwyer, D. Harlow, and D. Franklin, "Programming digital stories: Computer science and engineering design in the science classroom," *Science and Children*, vol. 53, no. 3, pp. 60–64, 2015. <https://www.researchgate.net/publication/283425066>
- [3] D. Franklin, P. Conrad, G. Aldana, and S. Hough, "Animal Tlatoque: Attracting middle school students to computing through culturally-relevant themes," in *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*, 2011, pp. 453–458. <https://pconrad.github.io/files/paper022.pdf>
- [4] A. Hansen, J. Gribble, A. Moran, D. Harlow, and E. Hansen, "Making computer science accessible," *Science and Children*, vol. 58, no. 5, pp. 80–85, 2021. <https://eric.ed.gov/?q=computer&id=EJ1296841>
- [5] National Center on Universal Design for Learning, "What is UDL?" 2013. [Web page]. Retrieved from <https://www.cast.org/impact/universal-design-for-learning-udl>
- [6] G. Salgarayeva, G. Ilyasova, A. Makhanova, and R. Abdrayimov, "The effects of using digital game based learning in primary classes with inclusive education," *European Journal of Contemporary Education*, vol. 10, no. 2, pp. 450–461, 2021. <https://doi.org/10.13187/ejced.2021.2.450>
- [7] R. E. Ladner and A. Stefik, "AccessCSforall: Making computer science accessible to K-12 students in the United States," *ACM SIGACCESS Accessibility and Computing*, no. 118, pp. 3–8, 2017. <https://doi.org/10.1145/3124144.3124145>
- [8] E. Schanzer, K. Fisler, and S. Krishnamurthi, "Bootstrap: Going beyond programming in after-school computer science," in *SPLASH Education Symposium*, 2013. <https://cs.brown.edu/~sk/Publications/Papers/Published/sfk-bootstrap-after-school/>
- [9] L. A. DeLyser, "A community model of CSforall: Analysis of community commitments for CS education," in *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE 2018)*, 2018, pp. 99–104. <https://doi.org/10.1145/3197091.3197142>
- [10] H. Bučková and J. Dostál, "Modern approach to computing teaching based on Code.org," *ICERI2017 Proceedings*, 2017, pp. 5091–5096. <https://doi.org/10.21125/iceri.2017.1337>
- [11] M. W. Ok, K. Rao, B. R. Bryant, and D. McDougall, "Universal design for learning in pre-K to grade 12 classrooms: A systematic review of research," *Exceptionality*, vol. 25, no. 2, pp. 116–138, 2017. <https://doi.org/10.1080/09362835.2016.1196450>
- [12] A. B. Mourão and J. F. Netto, "Inclusive model for the development and evaluation of accessible learning objects for graduation in computing: A case study," in *2018 IEEE Frontiers in Education Conference (FIE)*, 2018, pp. 1–8. <https://doi.org/10.1109/FIE.2018.8659129>
- [13] C. M. S. De Macedo and V. R. Ulbricht, "Accessibility guidelines for the development of learning objects," *Procedia Computer Science*, vol. 14, pp. 155–162, 2012. <https://doi.org/10.1016/j.procs.2012.10.018>
- [14] H. Abdellaoui, M. A. Ben Mohamed, K. Bacha, and M. Zrigui, "Ontology based description of an accessible learning object," in *Fourth International Conference on Information and Communication Technology and Accessibility (ICTA)*, Hammamet, Tunisia, 2013, pp. 1–5. <https://doi.org/10.1109/ICTA.2013.6815284>

- [15] J. T. Nganji and M. Brayshaw, "Personalizing learning materials for students with multiple disabilities in virtual learning environments," in *Proceedings of the 2015 Science and Information Conference (SAI)*, 2015, pp. 69–76. <https://doi.org/10.1109/SAI.2015.7237128>
- [16] C. Shelton, "How can we make computing lessons more inclusive?" in *Tomorrow's Learning: Involving Everyone. Learning with and about Technologies and Computing*, A. Tatnall and M. Webb, Eds., WCCE 2017, *IFIP Advances in Information and Communication Technology*, Springer, Cham, 2017, vol. 515, pp. 506–514. <https://doi.org/10.1007/978-3-319-74310-3>
- [17] R. E. Ladner and M. Israel, "'For all' in 'computer science for all,'" *Commun. of the ACM*, vol. 59, no. 9, pp. 26–28, 2016. <https://doi.org/10.1145/2971329>
- [18] P. King Sears, "Introduction to learning disability quarterly special series on universal design for learning: Part one of two," *Learning Disability Quarterly*, vol. 37, no. 2, pp. 68–70, 2014. <https://doi.org/10.1177/0731948714528337>
- [19] O. E. Hatlevik and K. A. Christophersen, "Digital competence at the beginning of upper secondary school: Identifying factors explaining digital inclusion," *Computers & Education*, vol. 63, pp. 240–247, 2013. <https://doi.org/10.1016/j.compedu.2012.11.015>
- [20] L. Florian, "Conceptualising inclusive pedagogy: The inclusive pedagogical approach in action," *Inclusive Pedagogy Across the Curriculum (International Perspectives on Inclusive Education)*, Emerald Group Publishing Limited, Leeds, vol. 7, pp. 11–24, 2015. <https://doi.org/10.1108/S1479-363620150000007001>
- [21] C. Shelton, "Beyond lesson recipes: First steps towards a repertoire for teaching primary computing," 2016. <https://core.ac.uk/download/pdf/74397184.pdf>
- [22] M. Israel, G. Jeong, M. J. Ray, and T. Lash, "Teaching elementary computer science through universal design for learning," in *Proceedings of the 51st ACM Technical Symposium on Computer Science Education (SIGCSE '20)*, 2020, pp. 1220–1226. <https://doi.org/10.1145/3328778.3366823>
- [23] M. Israel, T. Lash, L. Bergeron, and M. J. Ray, "Planning K-8 computer science through the UDL framework," 2018. <https://api.semanticscholar.org/CorpusID:218914082>
- [24] N. Lytle *et al.*, "Use, modify, create: Comparing computational thinking lesson progressions for STEM classes," in *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '19)*, 2019, pp. 395–401. <https://doi.org/10.1145/3304221.3319786>
- [25] I. Lee, F. Martin, and K. Apone, "Integrating computational thinking across the K-8 curriculum," *ACM Inroads*, vol. 5, no. 4, pp. 64–71, 2014. <https://doi.org/10.1145/2684721.2684736>
- [26] K. McGillivray, "The four layers of programming skills," 2018. [https://dev.to/kev\\_mcg/the-four-layers-of-programming-skills](https://dev.to/kev_mcg/the-four-layers-of-programming-skills)
- [27] G. Salgarayeva, G. Kopeeva, A. Kaptagayeva, and A. Yusupova, *Computer Science. Textbook for Students of the 6th Grade of the Secondary School*. Nur-Sultan: Arman-PV, 2020.
- [28] G. Salgarayeva, A. Makhanova, and L. Rsalina, *Computer Science. Textbook for Students of the 7th Grade of the Secondary School*. Nur-Sultan: Arman-PV, 2021.
- [29] A. Kadirkulov, A. Ryskulbekova, and G. Nurmukhanbetova, *Computer Science. Textbook for Students of the 7th Grade of the Secondary School*. Almaty: Almatykitap, 2023.
- [30] S. Mukhambetzhanova, A. Ten, and L. Demidova, *Computer Science. Textbook for Students of the 7th Grade of the Secondary School*. Almaty: Atamura, 2021.
- [31] S. Jacques, A. Ouahabi, and Z. Kanetaki, "Post-COVID-19 education for a sustainable future: Challenges, emerging technologies and trends," *Sustainability*, vol. 15, no. 8, p. 6487, 2023. <https://doi.org/10.3390/su15086487>

## 8 AUTHORS

**Gulnaz Salgarayeva** is a Candidate of Technical Sciences, is an Acting Professor of Kazakh National Women's Teacher Training University. She is a Director of the Institute of Physics, Mathematics and Digital Technologies. Her research interests focus on digital technologies in the educational process, inclusive education (E-mail: [salgara.g@qyzpu.edu.kz](mailto:salgara.g@qyzpu.edu.kz); ORCID: <https://orcid.org/0000-0001-9477-6245>).

**Aigul Makhanova** is Doctoral student at the Educational Program 8D01502-Computer Science, Kazakh National Women's Teacher Training University. Her research focuses on teaching Computer Science to students with special educational needs (E-mail: [makhanova.a@qyzpu.edu.kz](mailto:makhanova.a@qyzpu.edu.kz); ORCID: <https://orcid.org/0000-0001-6420-5249>).