A Programming Disposition Scale for High School Students

Chiu-Fan Hu[1,3], Yu-Tzu Lin[1,3], Cheng-Chih Wu[1,3*] and Hsueh-Chih Chen[2,3]
[1]Graduate Institute of Information and Computer Education, National Taiwan Normal University, Taiwan //
[2]Department of Educational Psychology and Counseling, National Taiwan Normal University, Taiwan //
[3]Institute for Research Excellence in Learning Sciences, National Taiwan Normal University, Taiwan //
chiufan@ntnu.edu.tw // linyt@ntnu.edu.tw // chihwu@ntnu.edu.tw // chcjyh@ntnu.edu.tw
[*]Corresponding author

(Submitted March 25, 2021; Revised September 3, 2021; Accepted September 4, 2021)

ABSTRACT: This study developed a scale to assess high school students' programming disposition. The scale was developed by utilizing a standardized test development process. The three constructs of the scale, namely confidence, persistence and flexible thinking, consisted of 9 items (3 items on each construct). Participants for the formal test of the scale were 1,332 students from 11 high schools. The validity and reliability of the programming disposition scale were validated via internal consistency, test-retest reliability, construct validity, discriminant validity, criterion-related validity, correlation coefficient of each subscale and confirmatory factor analysis. The analysis results showed that this scale is valid and reliable. The scale can serve as an assessment tool to assist teachers to instruct students learning programming, and help students determine whether taking programming courses in high school or pursuing programming-related majors in university. The effects of individual differences on programming disposition were also discussed to provide feasible educational implications.

Keywords: Disposition, Programming, Assessment tool, High school students

1. Introduction

A disposition is a tendency to display particular behaviors in a certain situation (Biber et al., 2013). Various patterns of thinking, such as confidence and attitude, enable one to be critical, thoughtful, and willing to work in a complex society (Wilkins, 2000). It includes not only students' confidence, curiosity, values and attitudes but also flexible thinking and the development of strategies for problem solving (Whitin, 2007). Students' inclinations and dispositions serve as predictors for their likelihood of taking related courses and pursuing various fields of study (Wilkins, 2000). The importance of student inclination and tendencies (disposition) has been previously addressed in the area of mathematics education. The National Council of Teachers of Mathematics (NCTM, 1989; NCTM, 2000) repeatedly stated the importance for teachers to improve and assess students' mathematical disposition. A positive disposition towards mathematics is considered to be more important than mathematical knowledge (Kusmaryono et al., 2019; Wilkins, 2000).

Programming is a subject related to mathematics and is considered as an integral component of K-12 curriculum as mathematics in many countries as it is a systematic way of approaching problem solving (Burrus & Moore, 2016; Winslow, 1996). In fact, programming has become an essential subject in K-12 schools to cope with the need of learning computational thinking (Lye & Koh, 2014). However, high school students often feel frustrated in learning text-based programming and have lower learning motivation (Galgouranas & Xinogalos, 2018). This would also affect students' academic intention for advanced study (Grandell et al., 2005). It might be beneficial if we can promote students' programming disposition. However, there is still less relevant research.

The development of assessment tools for disposition is still an open problem. To assess one's mathematical disposition, several tools have been developed to identify students' beliefs and attitudes (Royster et al., 1999), confidence (Wilkins, 2000), persistence (Breen et al., 2010), and flexible thinking (Whitin, 2007). Regarding programming, there is no assessment tool for disposition. In fact, it seems more challenging to develop a valid tool for assessing programming disposition because programming involves more various knowledge (e.g., programming syntax, constructs, and computer architecture) and skills (e.g., the use of IDEs, coding, and debugging). As Tsai et al. (2019) indicated, there is a lack of assessment tools for programming disposition in high school.

To fill research gaps, this study aims to develop a standardized scale to assess high school students' programming disposition. The disposition was assessed in terms of students' confidence, persistence, and flexible thinking on learning programming. Accordingly, the following research questions were explored:

Q1. Is the proposed programming disposition scale a valid and reliable assessment tool?
Q2. Does the second-order model of programming disposition show a good goodness of fit?

2. Literature review

2.1. Programming learning

Learning programming skills is often seen as difficult (Fitzgerald et al., 2008; Rum & Ismail, 2017; Sáez-López et al., 2016). The difficulties often deal with the abstract nature of programming (Bennedsen & Caspersen, 2006), intensive problem solving (Yurdugül & Aşkar, 2013), and using complex hierarchy of skills (Gray et al., 1993). Many studies have discussed that reducing the learning difficulties may also be linked with various attitudinal issues rather than intrinsic complexity of programming, such as complexity of syntax and algorithms (Hu et al., 2020; Luxton-Reilly, 2016). The idea of supporting and developing positive attitude in students has received considerable attention in programming education. Hu et al. (2021) advocated that programming instruction should emphasize arousing students' interests and improving attitudes rather than developing complex knowledge and skills only. Previous research has suggested that it is essential to develop K-12 students' dispositions in a curriculum (Katz, 1993). Students with a positive disposition have a curiosity in learning, appreciate the usefulness of learning subjects, are more confident of problem solving, and consequently, they are more disposed to apply their ability (Kusmaryono et al., 2019). What is more, without proper instruction to arouse students' disposition, students might have a negative disposition in learning. Students' attitudes towards programming have been investigated from various perspectives, such as self-efficacy (Sun & Hsu, 2019; Tsai et al., 2019), confidence in programming skills (Eliasson et al., 2006), and persistence of long-term learning (Eliasson et al., 2006; Gomes et al., 2012). However, there are few studies targeted on investigating students' programming dispositions. In addition, computer science educators are concerned about the lack of readily available, validated, or standardized assessment instruments in the field (Margulieux et al., 2019; Tew & Dorn, 2013). A rigorous process to develop the instruments is needed.

Besides programming disposition, there are still other factors that affect students' learning of programming, such as gender (Baser, 2013; Kong et al., 2018; Master et al., 2016), mathematical skills and abilities (Burrus & Moore, 2016; Erümit, 2020), science learning (Durak & Saritepeci, 2018), and parental support (Mason & Rich, 2020; Master et al., 2017). These factors might also affect students' programming disposition.

2.2. Construct of programming disposition

Student's programming abilities are correlated with their mathematical skills (Byrne & Lyons, 2001). The training of logical and abstract thinking, and reasoning in mathematics are relevant to working with abstract concepts and symbol manipulation in programming (Pioro, 2006). Students' mathematical dispositions served as a major foundation and springboard in our developing the construct of programming disposition. The NCTM (2000, see Table 1) has described students' dispositions as being relevant to their efforts in solving difficult problems and observing complex patterns, regularities, and correlations; these dispositions include confidence, perseverance, flexible thinking, and curiosity (NCTM, 2000; Whitin, 2007). Programming has been found as an effective tool for practicing computational thinking (Grover & Pea, 2013). The disposition towards computational thinking proposed by International Society for Technology in Education and the Computer Science Teachers Association (ISTE & CSTA, 2011, see Table 1) is also included as an important reference. The reference of NCTM and ISTE/CSTA constructs, along with literature in learning programming, allowed the construction of programming disposition scale to focus upon confidence, perseverance, and flexible thinking. The arguments are provided below.

Table 1. Constructs of mathematics/computational-thinking disposition

Mathematics disposition NCTM (2000)	Computational thinking disposition ISTE/CSTA (2011)
Confidence	Confidence in dealing with complexity
Perseverance	Persistence in working with difficult problems
Flexible thinking	Tolerance for ambiguity
Curiosity	Ability to deal with open-ended problems
	Ability to communicate and work with others to achieve the goal

Students' confidence and persistence (or perseverance) are both identified by NCTM and ISTE/CSTA as being important factors. Individual's confidence in dealing with complex problems is an important personal trait for

learning computer programming. Golding's et al. (2006) study has found that confidence was the most significant factor affecting one's performance in learning programming. There was a significantly positive correlation between students' confidence and their achievements in learning programming (Anastasiadou & Karakos, 2011; Baser, 2013). A student's level of confidence was found to be a major factor involved with the mastery of programming and especially for novices when trying to solve a complex problem (Eliasson et al., 2006).

Persistence, in terms of educational research, has been explained by many as a kind of continuously learning--one's tendency to pursue academic objectives (Pérez, 2018). In programming, persistence refers to continuing engagement when performing a challenging task. Persistency is needed to become a good programmer (Cheah, 2020; Jiau et al., 2009). Charlton and Birkett (1999) revealed that persistence is a predictor of programming achievement. Gomes et al. (2012) found persistence as being the most important reason students increase their performance in a programming course. Katz et al. (2006) also have found that students' persistence in programming correlated strongly with their grades. Perseverance (delineated by NCTM) has a very similar meaning with persistence applies to success in tackling difficult problems.

Flexible thinking has been characterized as the ability to restructure and transfer one's knowledge; that is, it enables people to understand, negotiate, and balance diverse views and beliefs-- those used to reach workable solutions (Barak & Levenberg, 2016). The process of learning programming does, indeed, involve such flexible thinking (Jang & Lew, 2014). One's personal flexibility is also an important characteristic in programming, such as approaching problems in multiple ways, being open to new ideas, and being open-minded (Begel & Nagappan, 2008). Concepts of flexible thinking include the disposition towards the following: "reflectivity, willingness to consider evidence contradictory to beliefs, willingness to consider alternative opinions and explanations, and a tolerance for ambiguity." This is also combined with a willingness to postpone closure (Stanovich & West, 1997). In this regard, the 'tolerance for ambiguity' is addressed in ISTE/CSTA and is a critical component of flexible thinking. The 'curiosity' delineated by NCTM is also a factor involved with flexible thinking. Students' exploratory attitudes and interests often manifest themselves with increased confidence while displaying flexibility and adaptability (Stokoe, 2012). These are aligned to concepts involved with flexible thinking.

The constructs relevant to 'ability' proposed by ISTE/CSTA were, additionally, removed because we focused on exploring students' programming dispositions (habits of mind) rather than their abilities (capabilities of doing something with knowledge and skills). Consequently, the scale utilized here consisted of three major constructs: confidence, persistence, and flexible thinking.

3. Method

We applied the standardized test development process to the development of the programming disposition scale used in this study. This development process involved two phases: (1) a pilot study and (2) a formal test. The pilot study was used to generate and analyze items. The formal test was used to examine the reliability and validity of the scale.

3.1. Participants

In the pilot study, convenience sampling was used to select 246 students (who did not participate in the formal test) from grades 10 to 12 who had learning experiences in programming from four Taipei high schools. In the formal study, the sample consisted of 117 (48%) tenth-grade, 76 (31%) eleventh-grade and 53 (22%) twelfth-grade students.

Table 2. The distribution of samples by school, grade, and academic track

Academic track	10^{th} grades	11^{th} grades		12^{th} grades		Total
	-	Science	Social science	Science	Social science	
Schools						
Tier 1	345	101	83	118	137	784
Tier 2	241	91	36	127	53	548
Total	586	192	119	245	190	1,332
		311		435		

Participants for the formal test of this study consisted of 1,332 students from 11 high schools in the Taipei metropolitan area in Taiwan. Stratified sampling was applied when recruiting the students. First, high schools were divided into two groups, Tier 1 and Tier 2, according to their traditional academic performance. Five to six schools were selected from each school group. Second, each school recruited one or two classes of students from each of the 10th to 12th grades. Finally, for 11th and 12th grades, both science and social science track students were recruited. High school students in Taiwan were divided into the two academic tracks after the 10th grade for their subject study. The distribution of samples by schools, grade, and academic track is shown in Table 2. All participants have programming experience because programming is covered in the 10th grade curriculum.

3.2. Procedure

The programming disposition scale was conducted on students in the formal test either by paper-and-pencil (two schools) or online (nine schools). The time for students to take the test was approximately 15 to 20 minutes.

3.3. Instruments

The programming disposition scale used here was developed based upon ones proposed by NCTM and ISTE/CSTA. The unique characteristics utilized in programming were considered when generating the constructs as discussed in section 2.2.

In the pilot study, draft items were adapted from various studies, such as "confidence" from the Fennema-Sherman Mathematics Attitudes Scales (Fennema & Sherman, 1976), "persistence" from (Breen et al., 2010), and "flexible thinking" from (Stanovich & West, 1997). Some items specifically related to programming aspects were added by the expert panel. A panel of seven experts included five computer science educators and two psychological and educational test professionals. They discussed and finalized 19 draft items (see Table 4) for further item analysis in the pilot study. Finally, a total of nine items were selected for the final scale used in the formal test. Three items were selected for each subscale (see Table 3). Item 6 is a negatively worded question which was reversed scored. The items developed here were selected based upon existing research, in which the scales used were mainly 5-point scales. Research by Croasmun and Ostrom (2011) has shown that a scale is both reliable and stable for both 4-point Likert and 5-point Likert scales. A 5-point Likert scale ranging from 1(strongly disagree) to 5 (strongly agree) was, thus, used in this study.

Table 3. Items of programming disposition scale

Constructs	Definition	Items
Confidence	Degree of having trust in programming	C1 I can get good grades in programming. C2 I can solve difficult programming tasks. C3 I believe I can learn programming.
Persistence	Continuing engagement in programming when facing a challenging task or spending a longtime to solve the task	P1 When presented with a difficult programming task, I increase my efforts. P2 I continue to work on a programming task even I have spent a long time to solve it and was not successful. P3 After learning programming for a while, I tend to give up.
Flexible thinking	Attempting to think differently or considering alternative solutions	FT1 I would try alternative solutions when solving problems similar to a previous one. FT2 I understand some programming tasks just cannot be solved in a short time. FT3 I consider alternative solutions when solving programming tasks.

Two instruments were used in this study to ensure the validity of the programming disposition scale. The Bebras Challenge (see https://www.bebras.org/) had over 2,872,000 students in 43 countries participated in 2019. The main goal of it is "to motivate pupils to be interested in informatics topics and to promote thinking which is algorithmic, logical, operational, and based on informatics fundamentals" (Dagienė & Stupuriene, 2016). The Bebras Challenge score was used to evaluate the correlation to the programming disposition scale in this study. The Comprehensive Assessment Program [CAP] for junior high school students is an examination for all 9th students in Taiwan. The examination scores play an important part for admitting students into secondary schools. CAP consists of Chinese, English, mathematics, natural science and social studies. This study used the CAP scores of mathematics and Chinese to assess the discriminant validity of programming disposition scale.

Additionally, three pieces of background information were collected from students, including gender (male, female), academic track (science, social science), and attitudes towards the degree of parental support (5-point Likert scale ranging from 1 to 5). This information was used to examine the construct validity of the programming disposition scale.

3.4. Data analysis

In the data analysis procedure, we analyzed data with SPSS 23.0 for Windows and LISREL 8.7 for Windows. Descriptive statistics were firstly performed to calculate the means, standard deviations and percentiles of student's programming disposition scores. Then, to test our research questions, the validity and reliability of this scale were evaluated using t tests and person correlation analysis to establish the internal consistency, test-retest reliability, criterion validity, discriminant validity and construct validity. A confirmatory factor analysis [CFA] was performed to identify the factor structure and items of the programming disposition scale. Independent t tests were used to examine the difference in gender and academic track. Pearson correlation analysis was conducted to test the correlations between parental support and programming disposition.

4. Results and discussion

4.1. Pilot study: Item analysis

The standards of evaluating included an improvement of internal consistency, item discrimination, factor loading, item-total correlation and individual item reliability. CFA results showed $\chi^2 = 769.18$ ($df = 149$), $p < .001$ and analysis of 19 items showed in Table 4.

Table 4. CFA results of 19 items

Construct	Item		Alpha if item deleted	Factor loading	Item-total correlation	Individual item reliability	*t*
Confidence	1.	I feel confident in programming.	.93	.85	.80**	.72	16.46***
	2.	I can get good grades in programming.[a]	.93	.79	.75**	.62	14.13***
	3.	I believe I can learn programming.[a]	.93	.82	.81**	.67	15.12***
	4.	I can solve difficult programming tasks.[a]	.93	.85	.82**	.72	16.33***
	5.	I cannot be good in programming.[b]	.93	.63	.62**	.40	11.43***
	6.	Programming is my worst learning activity.[b]	.93	.53	.55**	.28	9.62***
Persistence	7.	When presented with a difficult programming task, I increase my efforts.[a]	.93	.84	.83**	.71	16.34***
	8.	I tend to give up after spending much time on a programming task.[b]	.94	.32	.38**	.10	5.18***
	9.	I continue to work on a programming task even I have spent a long time to solve it and was not successful.[a]	.93	.85	.84**	.73	15.71***
	10.	I commit to spend a longtime to learn programming.	.93	.82	.81**	.67	17.03***
	11.	I believe learning programming requires a longtime effort.	.93	.46	.50**	.21	7.12***
	12.	After learning programming for a while, I tend to give up.[a b]	.93	.47	.54**	.22	9.29***
Flexible thinking	13.	I would try alternative solutions when I encountered difficulty in solving a programming task.	.93	.87	.83**	.76	17.37***
	14.	I always formulate solutions clearly before jumping into coding.	.93	.61	.63**	.37	9.47***
	15.	I would try alternative solutions when solving problems similar to a previous one.[a]	.93	.84	.77**	.70	14.35***

16. I understand some programming tasks just cannot be solved in a short time.[a]	.93	.60	.60**	.36	9.48***
17. I consider alternative solutions when solving programming tasks.[a]	.93	.84	.78**	.71	14.32***
18. I try to find out other solutions if I cannot solve a programming task.	.93	.82	.77**	.67	13.60***
19. I understand that not all problems can be solved by programming.	.94	.14	.13*	.02	1.78

Note. $^{*}p < .05$. $^{**}p < 0.01$. $^{***}p < 0.001$.[a] The item was included in the final programming disposition scale. [b] The item was a negative item.

First, according to the values of alpha if item deleted, each item was reliable (whole scale α = .93). The t-tests results of high and low scoring groups showed items had high discrimination (excluding item 19). The factors loaded between .14 and .87. Item 8 and 19 factor loading < .45. Further, the individual item reliability was between 0.02 and 0.76. 8 items (item 5, 6, 8, 11, 12, 14, 16 and 19) were considered to be deleted (individual item reliability < 0.5). The results of Pearson correlation showed that a significant correlation between each item and whole scale.

According to the results, item 1, 2, 3, 4, 7, 9, 15, 17 were included. In this scale, persistence means continuing engagement in programming when facing a challenging task or learning for a while. Compare to the other items, item 12 clearly states "after learning programming for a while," which could reflect the point in the persistence concept, "continuously for a while." As a result, we selected item 12 in the item pool. The concepts of flexible thinking include attempting to think carefully, considering alternative solutions and having a tolerance for ambiguity. The statement in item 16, "some programming tasks could not be solved soon" means that subjects needed to think more carefully or consider other possibilities, which was a kind of ambiguity. So item 16 was included. Finally, there were three items for each subscale. In the confidence subscale, item 1 to 4 were suggested to be included. However, concepts contained in item 2 to 4 already were enough to reflect item 1, in addition, to ensure the consistency in three subscales, we deleted item 1. Finally, the programming disposition scale was composed of 9 items.

According to the results of item analysis, the values of the goodness of fit were examined. The results found that $\chi^2 = 60.25$ (df = 24), $p < .001$, GFI = .95, AGFI = .90, RMR = .04, RMSEA = .07, NFI = .98, RFI = .96, CFI = .98, PGFI = .52, PNFI = .65, CN = 160.59. The results showed that the values of the goodness of fit are good.

4.2. Reliability and validity (Q1)

Table 5 shows the descriptive statistics of student's programming disposition in the formal test. The mean score for all participants was 28.45, averaged 3.22 for each item. Overall, students' programming disposition was found to be "medium" to "high." Students displayed the highest scores in flexible thinking (M = 10.08). Intermediate was that of persistence (M = 9.37), while confidence (M = 8.97) was shown to be the lowest. Our result with regard to "confidence" was similar to the TIMSS (Trends in International Science and Mathematics, 2020) study which showed that Taiwanese students lacked confidence in science and mathematics, although their performance has been shown to be higher than most of the countries (TIMSS, 2020).

Table 5. Descriptive statistics of student's programming disposition (N = 1,332)

		Total	Confidence	Persistence	Flexible thinking
M		28.42	8.97	9.37	10.08
SD		6.57	2.49	2.43	2.34
Min		9.00	3.00	3.00	3.00
Max		45.00	15.00	15.00	15.00
Percentiles	10	20.00	6.00	6.00	7.00
	20	23.00	7.00	7.60	8.00
	30	25.00	8.00	8.00	9.00
	40	28.00	9.00	9.00	10.00
	50	29.00	9.00	10.00	10.00
	60	30.00	10.00	10.00	11.00
	70	32.00	10.00	11.00	12.00
	80	34.00	11.00	11.00	12.00
	90	36.00	12.00	12.00	13.00

Cronbach's coefficient alpha (α) was used to test the internal consistency of the scale. The Cronbach's α of the entire scale was found to be .91. The subscales for confidence, persistence and flexible thinking were found to be .83, .78, and .78 respectively. The correlation coefficient for test–retest reliability was found to be .89 for the scale, and .86, .77, and .77 for the subscales of confidence, persistence, and flexible thinking, respectively. The correlations between each subscale are given in Table 6. The correlation coefficients are between .70 and .74. There is a positive correlation between each subscale. These results showed that this scale is reliable.

Table 6. Correlation coefficient of subscale

Subscale	*n*	*M*	*SD*	1	2	3
1. Confidence	1,332	8.97	2.49	1		
2. Persistence	1,332	9.37	2.43	.74**	1	
3. Flexible Thinking	1,332	10.08	2.34	.70**	.74**	1

Note. $^{**}p < .01$.

Bebras Challenge scores from 30 students were used to evaluate the criterion-related validity of the scale. The Bebras Challenge test, based on informatics fundamentals, is a context for understanding students' computational thinking. To solve Bebras Challenge tasks, students need to demonstrate their ability to understand informatics fundamentals. They accomplish this by using information computation, data processing, data visualization, algorithm and programming concepts (Dagienė & Futschek, 2008). Our analysis showed a positive correlation between students' Bebras Challenge performance and their programming disposition scale ($r = .48$; $p < .01$). This result was in agreement with findings by Araujo et al. (2017) and arguing that the Bebras Challenge performance test was a good measure of students' aptitudes in computer science (Combéfis & Stupurienė, 2020). Therefore, programming dispositions correlates with computer science learning.

The construct validity of the scale shows that students' programming dispositions were accurately reflected and consistent with previous research findings and is consistent with respect to gender differences, academic track, and parental support (as cited in the previous sections). Gender differences with respect to programming dispositions are described as follows. Table 7 shows that male students ($M = 30.19$, $SD = 6.34$) had a higher programming disposition ($t = 8.32$; $p < .001$) than female students ($M = 27.22$, $SD = 6.45$). The result is consistent with the findings of previous studies which show that male students display more positive attitudes towards programming (Kong et al., 2018; Master et al., 2016). Male students, additionally, also displayed higher confidence, persistence and flexible thinking than did their female counterparts. This is consistent with previous research in computer science with respect to gender differences. Male students also had higher levels of confidence when encountering more difficult programming problems than female students (Settle et al., 2015). Katz et al. (2006) also showed that male students had a higher persistence in executing programming tasks than females. It is, consequently, important that these gender differences can be identified so that additional strategies can be developed to improve students' programming disposition: addressing the needs of both male and female students.

Table 7. Gender and programming disposition

Construct	Male		Female		*t*(744)	*p*	Cohen's *d*
	M	*SD*	*M*	*SD*			
Programming disposition	30.19	6.34	27.22	6.45	8.32	.000	0.33
Confidence	9.66	2.47	8.51	2.4	8.49	.000	0.33
Persistence	9.93	2.39	8.99	2.39	7.04	.000	0.28
Flexible thinking	10.61	2.21	9.72	2.35	6.91	.000	0.28

Another important variable to consider when examining the validity of this scale is that academic track of the individual student. Table 8 shows that students enrolled in a science track ($M = 30.26$, $SD = 6.42$) had a significantly higher programming disposition score ($t = 9.55$; $p < .001$) than students in a social science track ($M = 25.63$, $SD = 6.69$). In Taiwan, high school students in grades 11 and 12 are divided into two academic tracks: science and social science. High school students in the science track often enroll in additional science and advanced math courses in grades 11 and 12. Students in the social science track, however, tend to enroll in more social studies, humanity, and intermediate math courses rather than additional science and math courses. In this study, we found students with science background had more positive programming dispositions in all three constructs: confidence, persistence, and flexible thinking. The findings support the idea that the learning of programming is strongly linked with mathematical skills and abilities (Burrus & Moore, 2016) and science subjects (Durak & Saritepeci, 2018).

With regard to the role of parental support, our findings (r = .35, p < .01) are consistent with previous studies that showed a positive correlation between the degree of parental support and programming dispositions. Previous studies have shown that the more the parents valued programming activities, the more positive were the students' attitudes (Mason & Rich, 2020; Master et al., 2017). In this study, we investigated the link between parental support and students' programming dispositions. Our findings reveal that parental support shows a very definite positive correlation with programming dispositions. These findings are consistent with the results in the 2018 Programme for International Student Achievement and the 2019 findings of the Organization for Economic Cooperation and Development (OECD, 2019). The more support students got from parents, the higher the dispositions.

Table 8. Academic track and programming disposition

Construct	Science		Social science		*t*(1330)	*p*	Cohen's *d*
	M	*SD*	*M*	*SD*			
Programming disposition	30.26	6.42	25.63	6.69	9.55	.000	.50
Confidence	9.59	2.50	8.06	2.47	8.31	.000	.44
Persistence	10.00	2.38	8.28	2.54	9.40	.000	.49
Flexible thinking	10.68	2.23	9.28	2.45	8.05	.000	.42

Table 9 shows that students' programming dispositions were positively correlated with CAP mathematics scores. The CAP Chinese scores were, however, shown to be consistently negative. This shows that the programming disposition scale has a high discriminant validity. Erümit (2020) has indicated that mathematical activities had a positive effect on thinking flexibly for solving programming problems and persistence of programming learning. Katz et al. (2006) also found that learning experience of relevant subjects affected students' persistence of programming learning. In fact, the fields of programming and mathematics involve similar cognitive processes, such as logical thinking, computing, reasoning and problem solving. Therefore, mathematical ability and learning experiences are correlated with students' confidence in learning programming, persistence in facing complex tasks and thinking flexibly.

Table 9. Discriminant validity (N = 1,332)

	Programming disposition	Confidence	Persistence	Flexible thinking
Mathematics	$.12^{**}$	$.11^{**}$	$.11^{**}$	$.12^{**}$
Chinese	$-.12^{**}$	$-.13^{**}$	$-.11^{**}$	$-.08^{*}$

Note. $^{**}p < .01$

4.3. Confirmatory factor analysis (Q2)

CFA was used to examine the values of the goodness of fit. The results found χ^2 = 201.04 (df = 24), p < .001 (Figure 1). The results of the CFA did not show a good statistical fit probably due to our large sample size (over 200). For this reason, other statistical analyses needed to be used (Rigdon, 1995). The measurement for the goodness of fit here is composed of absolute fit indexes, relative fit indexes, and parsimonious fit indexes (Bagozzi & Yi, 1988). The analysis of this scale is: Absolute fit index: GFI = .97, AGFI = .94, RMR = .03, RMSEA = .07; relative fit index: NFI = .99, RFI = .98, CFI = .90; parsimonious fit indexes: PGFI = .52, PNFI = .66, CN = 312.84. This model passed all 10 standards (Table 10). In addition, the factor loading of all items were higher than the acceptable level (ranged from 0.57 to 0.86) (Figure 1). Item C2 (I can solve difficult programming tasks), P1 (When presented with a difficult programming task, I increase my efforts), FT1 (7. I would try alternative solutions when solving problems similar to the previous one) had the highest factor load in confidence, persistence and flexible thinking respectively. Table 11 shows the composite reliability [CR] > .7, average variance extracted [AVE] > 0.5. These results revealed this model was confirmed and produces high reliability and validity. The CFA supported that the construct of programming disposition is composed of confidence, persistence, and flexible thinking. "Confidence" among three constructs has the highest CR and AVE.

Figure 1. CFA diagram of the scale

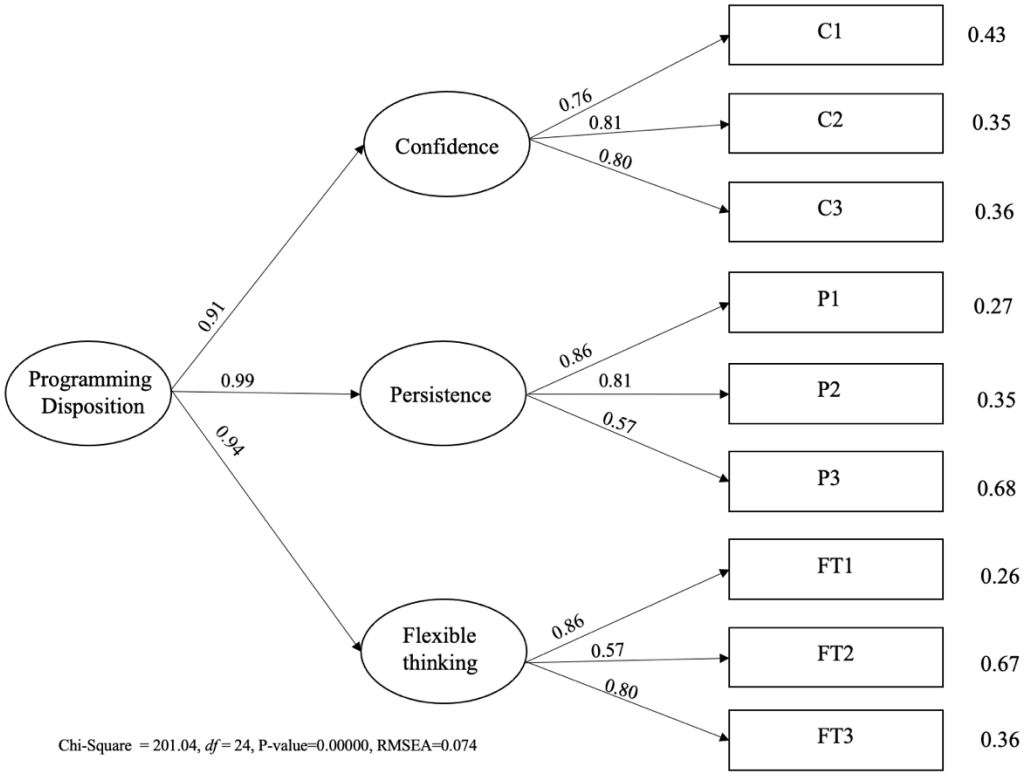

Table 10. Values of goodness of fit index

Goodness of fit index	Second-order factor analysis	Fit criteria of goodness of fit
Absolute fit index		
$\chi 2$	201.04	-
Df	24	-
GFI	0.97	> 0.9, good fit value
AGFI	0.94	> 0.9, good fit value
RMR	0.03	< 0.05, good fit value
RMSEA	0.07	< 0.08, reasonable fit value
Relative fit index		
NFI	0.99	> 0.9, good fit value
RFI	0.98	> 0.9, good fit value
CFI	0.99	> 0.9, good fit value
Parsimonious fit indexes		
PGFI	0.52	> 0.5
PNFI	0.66	> 0.5
CN	312.84	> 200 good sample quantity

Table 11. CR and AVE of three construct

Construct	CR	AVE
Confidence	0.83	0.62
Persistence	0.79	0.57
Flexible thinking	0.79	0.57

5. Significance of the programming disposition scale in educational settings

The significance of programming instruction has been addressed in literature (Burrus & Moore, 2016; Winslow, 1996). Previous findings have indicated that many students struggle with computer programming, which affects their engagement and motivation (Chookaew et al., 2015; Eliasson et al., 2006). Programming has a different nature from other disciplines because it involves both syntactic details and complex problem solving processes,

which requires intensive flexible thinking and persistence. Programming dispositions not only describes how much students are confident in programming, but also how they confront complicated problems. It also prescribes students' temperament of their roles when engaged in task performance (Association for Computing Machinery & IEEE Computer Machinery, 2020). Although previous research has devoted to studying effective instructional strategies for programming, it still lacks deeper exploration about students' disposition. Our research contributes to reveal more about students' behaviors and attitudes that characterize the inclination to carry out programming tasks.

The proposed programming disposition scale is an instrument for exploring how students communicate with programming tasks and their willingness to reflect on their own thinking and problem solving during programming. In educational settings, the programming disposition scale could be an effective tool for teachers to understand students' learning and evaluate the effects of instruction. Since the disposition scale moderates the behavior of applying knowledge and skills that becomes the context where and why the knowledge and skills are applied (Kusmaryono et al., 2019). This can, thus, be used as a guide for teachers to develop adaptive instruction to inspire and motivate their students for future studies or careers. A more inclusive learning environment can also be developed for students with different genders or from different cultures. In addition, teachers can evaluate whether their instructional strategies would inspire students' programming disposition, e.g., they can improve the implementation of STEM (Science, Technology, Engineering, Mathematics) education by considering programming disposition to arouse students' awareness of the integration of computational thinking and STEM disciplines.

6. Educational implications and suggestions

On the basis of research results, we produced some recommendations as follows.

Adaptive instruction: Among the three constructs, students' confidence was the lowest. This finding was similar to Mathematics. TIMSS (2020) released "TIMSS 2019 International Results in Mathematics and Science." The report showed that students in Taiwan often lack confidence in Mathematics and Science. Prior studies have proved that improper instructional design might lead to negative disposition (Katz, 1993). Therefore, instead of lecture-based instruction, more adaptive instruction should be provided based on students' characteristics. Exploration and experiment activities are effective for enhancing K-12 students' persistence and confidence. Through the process of struggling with complex problems, students' problem solving abilities can also be improved. Regarding gender issues, more adaptive learning activities should be designed to target to arouse females' interests and dispositions in programming. For example, Dagienė et al. (2015) found that through the task of dance moves, female students could understand better about the instructions or algorithm steps. Proper programming tools, such as visual programming, is also effective for engaging more females in programming (Baytak & Land, 2011; Kelleher & Pausch, 2006).

STEM instruction: The analysis of academic track showed the learning experience and ability of science and mathematics had a correlation with programming disposition. The knowledge and skills of science and mathematics should be integrated with programming practices. Lin et al. (2021) suggested that STEM education from multidisciplinary, such as programming and science, would increase students' interests. Erümit's (2020) study also pointed integrating mathematical activities into programming learning practices had a positive effect on thinking flexibly for solving programming problems and persistence of programming learning. Lin et al. (2019) found that through the STEM instruction, students had a higher confidence on programming learning. The results reveal "STEM" is an effective instructional strategy.

Jigsaw cooperative learning: Jigsaw cooperation is also an effective strategy for programming instruction. Teachers systemically divide learning tasks into different sub-tasks and assign students into groups, and each of the groups should complete one of the sub-tasks. Existing research has proved the effectiveness of Jigsaw strategies on students' knowledge building and confidence in programming (Garcia, 2021).

Parental support: Parental support has also been shown to be a vital factor in helping develop a student's programming disposition. It is imperative that schools help parents in understanding the importance of learning programming, a vital need for students' career development. Parents would not learn programming knowledge and skills but need to understand impacts of computing on daily life. The K-12 computer science framework listed three dimensions of impacts of computing: culture, social interaction and safety, law, and ethics. Thus, schools should conduct activities for parents to demonstrate the effects of computing, such as new cultural practices, equity and access to computing (K-12 Computer Science Framework Steering Committee, 2016).

7. Conclusion

This study develops a programming disposition scale for high school students. The scale is a five-point Likert-type scale and consists of 9 items. The internal consistency of the scale is excellent and the test-retest reliability is high. This scale appears to be respectably stable over time. The correlation coefficient of each subscale is positive. For the criterion validity, the scale shows a positive correlation with the Bebras Challenge. This scale also establishes the discriminant validity relevant to students' performance on mathematics and Chinese. The construct validity is validated by testing the variables of gender, academic track, and support from parents. The scale model has been verified by the CFA results. The structural equation modelling supports that the construct of programming disposition is composed of confidence, persistence, and flexible thinking. The results of these statistical analysis show that the scale is a valid and reliable tool. This study helps to expand our knowledge with respect to programming disposition, and improves the quality of assessment in programming education.

Our programming disposition scale has some strengths, such as filling an important gap in the field of assessment development for computer science education; teachers can utilize this scale to assess students' programming dispositions and find ways to help students learn in a programming course; students may, additionally, refer to the results on this scale and glean insights as to whether they should enroll in programming courses in high school or whether to choose related majors in a university setting; while this scale is primarily developed for students having experience in programming, it may also be useful for students who have little or no programming experience. However, a selection bias in participant recruitment might pose a threat to the internal validity of the research. All participants are in the Taipei metropolitan area in Taiwan. More subjects must be included in the future to get more generalized results for extending to other populations.

Acknowledgement

This research was sponsored by the Ministry of Science and Technology, Taiwan under Grant no. MOST 104-2511-S-003-055-MY3. We thank Dr. Ching-Lin Wu (Associate Professor at National Taiwan Normal University) for his assistance on this study.

References

Association for Computing Machinery & IEEE Computer Machinery. (2020). *Computing Curricula 2020: Paradigms for Global Computing Education*. https://www.acm.org/binaries/content/assets/education/curricula-recommendations/cc2020.pdf

Anastasiadou, S. D., & Karakos, A. S. (2011). The Beliefs of electrical and computer engineering students' regarding computer programming. *The International Journal of Technology, Knowledge and Society, 7*(1), 37-51. https://doi.org/10.18848/1832-3669/CGP/v07i01/56170

Araujo, A. L. S. O., Santos, J. S., Andrade, W. L., Guerrero, D. D. S., & Dagienė, V. (2017). Exploring computational thinking assessment in introductory programming courses. *Proceedings of 2017 IEEE Frontiers in Education Conference* (pp. 1-9). IEEE. https://doi.org/10.1109/FIE.2017.8190652

Bagozzi, R. P., & Yi, Y. (1988). On the evaluation of structural equation models. *Journal of the Academy of Marketing Science, 16*(1), 74-94.

Barak, M., & Levenberg, A. (2016). Flexible thinking in learning: An Individual differences measure for learning in technology-enhanced environments. *Computers & Education, 99*, 39-52. https://doi.org/10.1016/j.compedu.2016.04.003

Baser, M. (2013). Attitude, gender and achievement in computer programming. *MiddleEast Journal of Scientific Research, 14*(2), 248–255. https://doi.org/10.5829/idosi.mejsr.2013.14.2.2007

Baytak, A., & Land, S. M. (2011). Advancing elementary-school girls' programming through game design. *International Journal of Gender, Science and Technology, 3*(1), 243–253.

Begel, A., & Nagappan, N. (2008). Pair programming: what's in it for me? *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement* (pp. 120-128). ACM. https://doi.org/10.1145/1414004.1414026

Bennedsen, J., & Caspersen, M. E. (2006). Abstraction ability as an indicator of success for learning object-oriented programming? *ACM SIGCSE Bulletin, 38*(2), 39-43. https://doi.org/10.1145/1138403.1138430

Biber, A. C., Tuna, A., & Incikabi, L. (2013). An Investigation of critical thinking dispositions of mathematics teacher candidates. *Educational Research, 4*(2), 109-117.

Breen, S., Cleary, J., & O'Shea, A. (2010). Measuring students' persistence on unfamiliar mathematical tasks. *Proceedings of the British Society for Research into Learning Mathematics, 30*(3), 19-24.

Burrus, J., & Moore, R. (2016). The Incremental validity of beliefs and attitudes for predicting mathematics achievement. *Learning and Individual Differences, 50*, 246-251. https://doi.org/10.1016/j.lindif.2016.08.019

Byrne, P., & Lyons, G. (2001). The Effect of student attributes on success in programming. *ACM SIGCSE Bulletin, 33*(3), 49-52. https://doi.org/10.1145/377435.377467

Charlton, J. P., & Birkett, P. E. (1999). An Integrative model of factors related to computing course performance. *Journal of Educational Computing Research, 20*(3), 237-257. https://doi.org/10.2190/BTG0-7VQK-6XD3-G4C4

Cheah, C. S. (2020). Factors contributing to the difficulties in teaching and learning of computer programming: A Literature review. *Contemporary Educational Technology, 12*(2), Article ep272. https://doi.org/10.30935/cedtech/8247

Chookaew, S., Wanichsan, D., Hwang, G. J., & Panjaburee, P. (2015). Effects of a personalised ubiquitous learning support system on university students' learning performance and attitudes in computer-programming courses. *International Journal of Mobile Learning and Organisation, 9*(3), 240-257. https://doi.org/10.1504/IJMLO.2015.074207

Combéfis, S., & Stupurienė, G. (2020). Bebras based activities for computer science education: Review and perspectives. In K. Kori & M. Laanpere (Eds.), *Lecture Notes in Computer Science: Vol. 12518. Informatics in Schools. Engaging Learners in Computational Thinking* (pp. 15–29). Springer. https://doi.org/10.1007/978-3-030-63212-0_2

Croasmun, J. T., & Ostrom, L. (2011). Using Likert-type scales in the social sciences. *Journal of Adult Education, 40*(1), 19-22.

Dagienė, V., & Futschek, G. (2008). Bebras international contest on informatics and computer literacy: Criteria for good tasks. In R.T. Mittermeir, M.M. Sysło (Eds.), *Lecture Notes in Computer Science: Vol. 5090. Informatics Education - Supporting Computational Thinking* (pp. 15–29). Springer. https://doi.org/10.1007/978-3-540-69924-8_2

Dagienė, V., & Stupuriene, G. (2016). Bebras--A Sustainable community building model for the concept based learning of informatics and computational thinking. *Informatics in Education, 15*(1), 25-44. https://doi.org/10.15388/infedu.2016.02

Dagienė, V., Pelikis, E., & Stupurienė, G. (2015). Introducing computational thinking through a contest on informatics: Problem-solving and gender issues. *Information Sciences, 73*, 55-63.

Durak, H. Y., & Saritepeci, M. (2018). Analysis of the relation between computational thinking skills and various variables with the structural equation model. *Computers & Education, 116*, 191-202. https://doi.org/10.1016/j.compedu.2017.09.004

Eliasson, J., Westin, L. K., & Nordstrom, M. (2006). Investigating students' confidence in programming and problem solving. *Proceedings of 36th Annual Frontiers in Education Conference* (pp. 22-27). IEEE. https://doi.org/10.1109/FIE.2006.322490

Erümit, A. K. (2020). Effects of different teaching approaches on programming skills. *Education and Information Technologies, 25*(2), 1013-1037. https://doi.org/10.1007/s10639-019-10010-8

Fennema, E., & Sherman, J. A. (1976). Fennema-Sherman mathematics attitudes scales: Instruments designed to measure attitudes toward the learning of mathematics by females and males. *Journal for research in Mathematics Education, 7*(5), 324-326. https://doi.org/10.2307/748467

Fitzgerald, S., Lewandowski, G., McCauley, R., Murphy, L., Simon, B., Thomas, L., & Zander, C. (2008). Debugging: Finding, fixing and flailing, a multi-institutional study of novice debuggers. *Computer Science Education, 18*(2), 93-116. https://doi.org/10.1080/08993400802114508

Galgouranas, S., & Xinogalos, S. (2018). jAVANT-GARDE: A Cross-platform serious game for an introduction to programming with Java. *Simulation & Gaming, 49*(6), 751-767. https://doi.org/10.1177/1046878118789976

Garcia, M. B. (2021). Cooperative learning in computer programming: A Quasi-experimental evaluation of Jigsaw teaching strategy with novice programmers. *Education and Information Technologies, 26*, 1-18. https://doi.org/10.1007/s10639-021-10502-6

Golding, P., Facey-Shaw, L., & Tennant, V. (2006). Effects of peer tutoring, attitude and personality on academic performance of first year introductory programming students. *Proceedings of 36th Annual Frontiers in Education Conference* (pp. 7-12). IEEE. https://doi.org/10.1109/FIE.2006.322662

Gomes, A. J., Santos, A. N., & Mendes, A. J. (2012). A Study on students' behaviours and attitudes towards learning to program. *Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education* (pp. 132-137). ACM. https://doi.org/10.1145/2325296.2325331

Grandell, L., Peltomäki, M., & Salakoski, T. (2005). High school programming—A Beyond-syntax analysis of novice programmers' difficulties. *Proceedings of the Koli Calling 2005 Conference on Computer Science Education* (pp. 17-24). ACM.

Gray, W. D., Goldberg, N. C., & Byrnes, S. A. (1993). Novices and programming: Merely a difficult subject (why?) or a means to mastering metacognitive skills? A Review of Soloway's & Spohrer's, Studying the Novice Programmer. *Journal of Educational Research on Computers*, *9*(1), 131-140.

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

Hu, C. F., Wang, A. T., Wu, C. C., & Lin, Y. T. (2020). Identifying learning concepts for the new 12-year basic education ICT curriculum: A Delphi survey. *Bulletin of Educational Research, 66*(1), 77-102. https://doi.org/10.3966/102887082020036601003

Hu, C. F., Wu, C.C., Lin, Y. T., & Yu, C. H. (2021). Development of a computational thinking test for high school students. *International Journal on Digital Learning Technology, 13*(1), 1-21. https://doi.org/10.3966/2071260X2021011301001

International Society for Technology in Education & the Computer Science Teachers Association (ISTE & CSTA). (2011). *Operational Definition of Computational Thinking for K–12 Education.* https://cdn.iste.org/www-root/Computational_Thinking_Operational_Definition_ISTE.pdf

Jang, I. O., & Lew, H. C. (2014). Case studies in thinking processes of mathematically gifted elementary students through Logo programming. *Work, 4*, 9.

Jiau, H. C., Chen, J. C., & Ssu, K.-F. (2009). Enhancing self-motivation in learning programming using game-based simulation and metrics. *IEEE Transactions on Education, 52*(4), 555-562. https://doi.org/10.1109/TE.2008.2010983

K-12 Computer Science Framework Steering Committee. (2016). *K-12 Computer science framework.* https://k12cs.org/wp-content/uploads/2016/09/K-12-Computer-Science-Framework.pdf

Katz, L. G. (1993). *Dispositions as educational goals* (ED363454). ERIC. https://files.eric.ed.gov/fulltext/ED363454.pdf

Katz, S., Allbritton, D., Aronis, J., Wilson, C., & Soffa, M. L. (2006). Gender, achievement, and persistence in an undergraduate computer science program. *ACM SIGMIS Database: The DATABASE for Advances in Information Systems, 37*(4), 42-57. https://doi.org/10.1145/1185335.1185344

Kelleher, C., & Pausch, R. (2006). Lessons learned from designing a programming system to support middle school girls creating animated stories. In J. Grundy & J. Howse (Eds.), *Proceedings of Visual Languages and Human-Centric Computing* (pp. 165–172). IEEE. https://doi.org/10.1109/VLHCC.2006.30

Kong, S. C., Chiu, M. M., & Lai, M. (2018). A Study of primary school students' interest, collaboration attitude, and programming empowerment in computational thinking education. *Computers & Education, 127*, 178-189. https://doi.org/10.1016/j.compedu.2018.08.026

Kusmaryono, I., Suyitno, H., Dwijanto, D., & Dwidayati, N. (2019). The Effect of mathematical disposition on mathematical power formation: Review of dispositional mental functions. *International Journal of Instruction, 12*(1), 343-356.

Lin, Y. T., Wang, M. T., & Wu, C. C. (2019). Design and implementation of interdisciplinary STEM instruction: Teaching programming by computational physics. *The Asia-Pacific Education Researcher, 28*(1), 77-91. https://doi.org/10.1007/s40299-018-0415-0

Lin, Y. T., Yeh, M. K. C., & Hsieh, H. L. (2021). Teaching computer programming to science majors by modelling. *Computer Applications in Engineering Education, 29*(1), 130-144. https://doi.org/10.1002/cae.22247

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12?. *Computers in Human Behavior, 41*, 51-61. https://doi.org/10.1016/j.chb.2014.09.012

Luxton-Reilly, A. (2016). Learning to program is easy. *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education* (pp. 284-289). ACM. https://doi.org/10.1145/2899415.2899432

Margulieux, L., Ketenci, T. A., & Decker, A. (2019). Review of measurements used in computing education research and suggestions for increasing standardization. *Computer Science Education, 29*(1), 49-78. https://doi.org/10.1080/08993408.2018.1562145

Mason, S. L., & Rich, P. J. (2020). Development and analysis of the Elementary Student Coding Attitudes Survey. *Computers & Education, 153,* Article 103898. https://doi.org/10.1016/j.compedu.2020.103898

Master, A., Cheryan, S., & Meltzoff, A. N. (2016). Computing whether she belongs: Stereotypes undermine girls' interest and sense of belonging in computer science. *Journal of Educational Psychology, 108*(3), 424-437. https://doi.org/10.1037/edu0000061

Master, A., Cheryan, S., Moscatelli, A., & Meltzoff, A. N. (2017). Programming experience promotes higher STEM motivation among first-grade girls. *Journal of Experimental Child Psychology, 160*, 92-106. https://doi.org/10.1016/j.jecp.2017.03.013

National Council of Teachers of Mathematics (NCTM). (1989). *Curriculum and evaluation standards for school mathematics*. https://www.nctm.org/Standards-and-Positions/More-NCTM-Standards/Curriculum-and-Evaluation-Standards-(1989)

National Council of Teachers of Mathematics (NCTM). (2000). *Principles and standards for school mathematics*. https://www.nctm.org/uploadedFiles/Standards_and_Positions/PSSM_ExecutiveSummary.pdf

Organization for Economic Cooperation and Development (OECD). (2019). *PISA 2018 Results (Volume II): Where All Students Can Succeed*. https://www.oecd.org/pisa/publications/pisa-2018-results-volume-ii-b5fd1b8f-en.htm

Pérez, A. (2018). A Framework for computational thinking dispositions in mathematics education. *Journal for Research in Mathematics Education, 49*(4), 424-461. https://doi.org/10.5951/jresematheduc.49.4.0424

Pioro, B. T. (2006). Introductory computer programming: Gender, major, discrete mathematics, and calculus. *Journal of Computing Sciences in Colleges, 21*(5), 123-129.

Rigdon, E. (1995). A Necessary and sufficient identification rule for structural equation models estimated. *Multivariate Behavioral Research, 30*, 359-383. https://doi.org/10.1207/s15327906mbr3003_4

Royster, D. C., Kim Harris, M., & Schoeps, N. (1999). Dispositions of college mathematics students. *International Journal of Mathematical Education in Science and Technology, 30*(3), 317-333. https://doi.org/10.1080/002073999287851

Rum, S. N. M., & Ismail, M. A. (2017). Metocognitive support accelerates computer assisted learning for novice programmers. *Educational Technology & Society, 20*(3), 170-181.

Sáez-López, J. M., Román-González, M., & Vázquez-Cano, E. (2016). Visual programming languages integrated across the curriculum in elementary school: A Two year case study using "Scratch" in five schools. *Computers & Education, 97*, 129-141. https://doi.org/10.1016/j.compedu.2016.03.003

Settle, A., Lalor, J., & Steinbach, T. (2015). Reconsidering the impact of CS1 on novice attitudes. In A. Decker (Eds.),*Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (pp. 229-234). ACM. https://doi.org/10.1145/2676723.2677235

Stanovich, K. E., & West, R. F. (1997). Reasoning independently of prior belief and individual differences in actively open-minded thinking. *Journal of Educational Psychology, 89*(2), 342–357. https://doi.org/10.1037/0022-0663.89.2.342

Stokoe, R. (2012). Curiosity, a condition for learning. *The International Schools Journal, 32*(1), 63-65.

Sun, J. C. Y., & Hsu, K. Y. C. (2019). A Smart eye-tracking feedback scaffolding approach to improving students' learning self-efficacy and performance in a C programming course. *Computers in Human Behavior, 95*, 66-72. https://doi.org/10.1016/j.chb.2019.01.036

Tew, A. E., & Dorn, B. (2013). The Case for validated tools in computer science education research. *Computer, 46*(9), 60-66. https://doi.org/10.1109/MC.2013.259

Trends in International Mathematics and Science Study and the Progress in International Reading Literacy Study International Study Center (TIMSS). (2020). *TIMSS 2019 International Results in Mathematics and Science*. https://timssandpirls.bc.edu/timss2019/international-results/

Tsai, M. J., Wang, C. Y., & Hsu, P. F. (2019). Developing the computer programming self-efficacy scale for computer literacy education. *Journal of Educational Computing Research, 56*(8), 1345-1360. https://doi.org/10.1177/0735633117746747

Whitin, P. E.(2007). The Mathematics survey: A Tool for assessing attitudes and dispositions. *Teaching Children Mathematics, 13*(8), 426-433. https://doi.org/10.5951/TCM.13.8.0426

Wilkins, J. L. (2000). Preparing for the 21st century: The Status of quantitative literacy in the United States. *School Science and Mathematics, 100*(8), 405-418. https://doi.org/10.1111/j.1949-8594.2000.tb17329.

Winslow, L. E. (1996). Programming pedagogy—A Psychological overview. *ACM Sigcse Bulletin, 28*(3), 17-22. https://doi.org/10.1145/234867.234872

Yurdugül, H., & Aşkar, P. (2013). Learning programming, problem solving and gender: A Longitudinal study. *Procedia-Social and Behavioral Sciences, 83*, 605-610. https://doi.org/10.1016/j.sbspro.2013.06.115