TDSB ICT Skills Continuum
for
Elementary Coding

Grades

This document was produced by the TDSB's Teaching and Learning with Technology team

Kevin Bradbeer
Wayne Loo
Shelley Lowry
Julie Millan
Andrew Schmitt

Table of Contents

Goals of the TDSB ICT Standards

The ***TDSB ICT Standards*** document is a framework for students, teachers, and administrators to utilize technology as a tool for teaching and learning for K to 12 students throughout the TDSB. With the overall goal of improving student achievement, the ICT Standards document is a guide to help teachers integrate ICT into The Ontario Curriculum, into teaching practice, and into the student's repertoire of skills in order to support and enhance continuous learning.

In recognition of varying experience levels with ICT integration among teachers, this ICT Standards document seeks to bridge the gap between what Marc Prensky calls "Digital Natives" - today's learners, and "Digital Immigrants" – many of today's teachers. This document offers multiple entry points to accommodate teachers who span the digital literacy continuum and suggests various ICT experiences to prepare our students for the future. This document prescribes skills and the corresponding tools to provide the students with learning opportunities to master skills at certain grade levels.

This ICT Standards document provides different methods of delivering The Ontario Curriculum so that the various learning styles of our students may be addressed. Differentiated instruction requires the assessment of student profiles including student interests, readiness for the curriculum topic, and learning preferences. Technology is of interest and a common means of communication for today's youth, and therefore, an excellent strategy for student choice and engagement. Integrating digital literacy within the teaching and learning of content, process, product, and environment provides our students with essential skills for work and life.

Click for a digital copy of the TDSB ICT Standards: Digital Learning for Kindergarten to Grade 12

TDSB ICT Standards for Coding

The TDSB ICT Standards for coding have been developed in conjunction with the TDSB ICT Standards Document. This is a framework for using technology as a tool for teaching and learning and is meant to be used in conjunction with the Ontario Curriculum documents. The ICT Standards for coding are not a stand-alone course of study. The TDSB ICT Standards are developed for grade divisions.

Kindergarten to Grade 2

- Students will understand, design, develop and test algorithms (e.g., create a list of instructions in sequence to complete a task) for a specific purpose (e.g., planting a seed, going home from school)

Grades 3 - 5

- Students will design and plan a program to animate a character or tell a story (e.g., make a character dance, create an interactive image)

Grades 6 - 8

- Students will design, develop, program and test a game (e.g., create a game in which the player guides a character through a maze using a mouse, program a character to move and jump across platforms when the arrow keys are pressed)

TDSB ICT Skills Continuum for Coding

The skills continuum for coding provides skills specific for each grade from Kindergarten to Grade 8. Teachers may choose to use these skills as a coding skills assessment, “Look Fors”, and/or to determine an entry point for curriculum and coding integration specific to a particular class or student.

This TDSB Skills Continuum for Elementary Coding is not an exhaustive or exclusive list of skills required for coding. The skills in this continuum are directly related to the ICT Standards for Coding for each grade.

While there are many resources that will lead students through a progression of programming skills that may far exceed the skills listed in this document, the purpose of this document is to show teachers how coding skills can be used as a tool for students to demonstrate their learning to meet

the Ontario curriculum expectations, for the appropriate grade level and subject area. Coding is taught concurrently with the curriculum and not treated as a separate discipline.

The TDSB Skill Continuum for Elementary Coding includes the following categories for each grade:

- Understanding & Creating Algorithms (Programming Concepts)
- Writing Programs
- Testing & Debugging Programs (Code Maintenance)

Kindergarten

Students will understand, design, develop and test algorithms for a specific purpose (e.g., create a list of instructions in sequence to complete a task) (e.g., planting a seed, going home from school)

Understanding & Creating Algorithms (Programming Concepts)

- Identify the variables required to complete a task (e.g., ingredients in a recipe, characters in a story)
- Understand that algorithms are a list of steps (e.g., baking instructions, how to plant a seed, the sequence of a story)
- Understand that actions can be represented by symbols (e.g., the up arrow means move forward)

Writing Programs:

- Follow a sequence of steps and decisions (algorithms) needed to solve simple problems

Testing & Debugging Programs (Code Maintenance)

- Identify any errors or problems by running through the steps (e.g. the jam was spread but there is nothing to spread it on)
- Fix any errors and problems, and re-run sequence of steps to verify fixes. (e.g. change the order of sequence so the bread is taken out and then the jam can be spread on it)

1 Grade One

Students will understand, design, develop and test algorithms for a specific purpose (e.g., create a list of instructions in sequence to complete a task) (e.g., planting a seed, going home from school)

Understanding & Creating Algorithms (Programming Concepts)

- List the variables required to complete a task (e.g., ingredients in a recipe, key events in a story)
- Understand that algorithms are a list of steps (e.g., baking instructions, how to plant a seed, the sequence of a story)

Writing Programs

- Follow a path by using symbols (e.g., directions a character must take to move along a path; move forward, back, left, right)
- Order the sequence of steps and decisions (algorithms) needed to solve simple problems

Testing & Debugging Programs (Code Maintenance)

- Identify any errors or problems by running through the steps (e.g. character ends up at a wall and can't move forward)
- Fix any errors and problems, and re-run sequence of steps to verify fixes. (e.g. include a turn so the character ends up at the right spot)

2 Grade Two

Students will understand, design, develop and test algorithms for a specific purpose (e.g., create a list of instructions in sequence to complete a task) (e.g., planting a seed, going home from school)

Understanding & Creating Algorithms (Programming Concepts)

- Predict how changing a variable will change the final outcome of an algorithm (e.g., changing ingredients in a recipe, changing key event in a story)

Writing Programs

- Describe the sequence of steps (algorithms) needed to solve simple problems

Testing & Debugging Programs (Code Maintenance)

- Identify any errors or problems by running through the steps (e.g. cannot spread bologna)

- Fix any errors and problems, and re-run sequence of steps to verify fixes. (e.g. place bologna on bread instead of spreading it)

3 Grade Three

Students will design and plan a program to animate a character or tell a story (e.g., make a character dance, create an interactive image)

Understanding & Creating Algorithms (Programming Concepts)

- Identify parts of an algorithm that are repeated multiple times (e.g., walk forward 20 times)
- Create an algorithm that has repeating steps and incorporate a loop to replace it (e.g., an object moves across the screen)

Writing Programs

- Translate an algorithm into computer code (program) that incorporates repetition (e.g., use Scratch to make a character dance)

Testing & Debugging Programs (Code Maintenance)

- Identify any errors or problems by frequently running the code at various stages of the coding process (e.g. the intended character doesn't move)
- Fix any errors and problems, and re-run program to verify fixes. (e.g. code applied to wrong sprite character)

4 Grade Four

Students will design and plan a program to animate a character or tell a story (e.g., make a character dance, create an interactive image)

Understanding & Creating Algorithms (Programming Concepts)

- Identify conditionals (e.g. is there a wall?) and how it is used in an “If statement” to determine an action (e.g., IF there is a wall THEN turn right, ELSE go straight)
- Create an algorithm that has “if statements” inside a loop (e.g., character moves forward if there is no wall, and if there is a wall, then turn right and move forward- keep repeating this sequence)

Writing Programs

- Translate an algorithm into a program that incorporates “if statements” inside a loop (e.g., automate an object moving around the screen)

Testing & Debugging Programs (Code Maintenance)

- Identify any errors or problems by frequently running the code at various stages of the coding process (e.g. character still tries to move forward if there is a wall)
- Fix any errors and problems, and re-run program to verify fixes (e.g. did not include a turn if there is a wall in front of the character)

5 Grade Five

Students will design and plan a program to animate a character or tell a story (e.g., make a character dance, create an interactive image)

Understanding & Creating Algorithms (Programming Concepts)

- Identify conditionals (e.g. is there a wall?) and how it is used in a “repetition statement” to determine how many times an action repeats (e.g., Repeat 4 times, if there is a wall then turn right and go straight)

- Create an algorithm that has “if statements” and “repetition statements” that responds to user input (e.g. an object follows a mouse around the screen)

Writing Programs

- Translate an algorithm into a program that incorporates “if statements” and “repetition statements” that responds to user input (e.g. follow the mouse pointer around the screen)

Testing & Debugging Programs (Code Maintenance)

- Identify any errors or problems by frequently running the code at various stages of the coding process (e.g. character isn’t moving when key is pressed)

- Fix any errors and problems, and re-run program to verify fixes. (e.g. did not assign a key for that movement direction)

6 Grade Six

Design, develop, program and test a game (e.g., create a game in which the player guides a character through a maze using a mouse, program a character to move and jump across platforms when the arrow keys are pressed)

Understanding & Creating Algorithms (Programming Concepts)

- Identify the objectives (output) in various games (e.g., how the game is won, how a game is lost)
- Identify what data (e.g., button) is used in various games to control the character (input) (e.g. spacebar = jump, mouse = pointer)

Writing Programs

- Design a game that has an objective and user-controlled input. (e.g., character moves through a maze by following the pointer of a mouse, catch falling items)
- Write a program to move a character to an objective (e.g., complete a maze)

Testing & Debugging Programs (Code Maintenance)

- Identify any errors or problems by frequently running the code at various stages of the coding process (e.g. game does not accumulate point totals)
- Fix any errors and problems, and re-run program to verify fixes. (e.g. forgot to add a counter c=c+1)

7 Grade Seven

Design, develop, program and test a game (e.g., create a game in which the player guides a character through a maze using a mouse, program a character to move and jump across platforms when the arrow keys are pressed)

Understanding & Creating Algorithms (Programming Concepts)

- Identify how the objective (output) can be affected by the input in various ways (processing) (e.g. user controls character and game ends if a wall is hit on its way to the destination)

Writing Programs

- Design a game that has an objective, user-controlled input and obstacles/challenges (e.g., move a character through a maze avoiding the walls and rocks)
- Write a program that moves a character and includes obstacles to the objective (e.g. completing an obstacle course)

Testing & Debugging Programs (Code Maintenance)

- Identify any errors or problems by frequently running the code at various stages of the coding process (e.g. character does not respond to obstacles)
- Fix any errors and problems, and re-run program to verify fixes. (e.g. did not include the obstacle to keep checking for collision, using a loop)

8 Grade Eight

Design, develop, program and test a game (e.g., create a game in which the player guides a character through a maze using a mouse, program a character to move and jump across platforms when the arrow keys are pressed)

Understanding & Creating Algorithms (Programming Concepts)

- Identify variables that have been included in a game (e.g., score, speed, name, date of birth)

Writing Programs

- Design a game that has an objective, user-controlled input, obstacles/challenges and variables (e.g. a character moves through a maze while collecting coins, quiz)

- Write a program that moves a character and includes obstacles and variables to the objective and variables (e.g. completing an obstacle course with a particular score)

Testing & Debugging Programs (Code Maintenance)

- Identify any errors or problems by frequently running the code at various stages of the coding process (e.g. character responds to one obstacle but not any others)

- Fix any errors and problems, and re-run program to verify fixes. (e.g. need to include multiple and separate code to check for each obstacle)

TDSB ICT Skills Continuum for Coding (Kindergarten to Grade 2)

TDSB ICT Standards for Coding: (Kindergarten to Grade 2)

- Understand, design, develop and test algorithms for a specific purpose (e.g., create a list of instructions in sequence to complete a task) (e.g., planting a seed, going home from school)

Kindergarten	Grade 1	Grade 2
1. Identify the variables required to complete a task (e.g., ingredients in a recipe, characters in a story)	1. List the variables required to complete a task (e.g., ingredients in a recipe, key events in a story)	1. Predict how changing a variable will change the final outcome of an algorithm (e.g., changing ingredients in a recipe, changing key event in a story)
2. Understand that algorithms are a list of steps (e.g., baking instructions, how to plant a seed, the sequence of a story)	2. Understand that algorithms are a list of steps (e.g., baking instructions, how to plant a seed, the sequence of a story)	2. Describe the sequence of steps (algorithms) needed to solve simple problems
3. Understand that actions can be represented by symbols (e.g., the up arrow means move forward)	3. Follow a path by using symbols (e.g., directions a character must take to move along a path; move forward, back, left, right)	3. Test a simple algorithm in order to complete a task (e.g., does the story make sense?)
4. Follow a sequence of steps and decisions (algorithms) needed to solve simple problems	4. Order the sequence of steps and decisions (algorithms) needed to solve simple problems	4. Identify any errors or problems by running through the steps (e.g. cannot spread bologna)
5. Identify any errors or problems by running through the steps (e.g. the jam was spread but there is nothing to spread it on)	5. Test a simple algorithm to complete a task (e.g., does the plant grow? does the story make sense?)	5. Fix any errors and problems, and re-run sequence of steps to verify fixes. (e.g. place bologna on bread instead of spreading it)
6. Fix any errors and problems, and re-run sequence of steps to verify fixes. (e.g. change the order of sequence so the bread is taken out and then the jam can be spread on it)	6. Identify any errors or problems by running through the steps (e.g. character ends up at a wall and can't move forward)	
	7. Fix any errors and problems, and re-run sequence of steps to verify fixes. (e.g. include a turn so the character ends up at the right spot)	

TDSB ICT Skills Continuum for Coding (Grades 3-5)

TDSB ICT Standards for Coding: Grades 3-5

- Design and plan a program to animate a character or tell a story (e.g., make a character dance, create an interactive image)

Grade 3	Grade 4	Grade 5
1. Identify parts of an algorithm that are repeated multiple times (e.g., walk forward 20 times) 2. Create an algorithm that has repeating steps and incorporate a loop to replace it (e.g., an object moves across the screen) 3. Translate an algorithm into computer code (program) that incorporates repetition (e.g., use Scratch to make a character dance) 4. Identify any errors or problems by frequently running the code at various stages of the coding process (e.g. the intended character doesn't move) 5. Fix any errors and problems, and re-run program to verify fixes. (e.g. code applied to wrong sprite character)	1. Identify conditionals (e.g. is there a wall?) and how it is used in an "If statement" to determine an action (e.g., IF there is a wall THEN turn right, ELSE go straight) 2. Create an algorithm that has "if statements" inside a loop (e.g., 3. Translate an algorithm into a program that incorporates "if statements" inside a loop (e.g., move an object around the screen 4. Identify any errors or problems by frequently running the code at various stages of the coding process (e.g. character still tries to move forward if there is a wall) 5. Fix any errors and problems, and re-run program to verify fixes (e.g. did not include a turn if there is a wall in front of the character)	1. Identify conditionals (e.g. is there a wall?) and how it is used in a "repetition statement" to determine how many times an action repeats (e.g., WHILE there is a wall then turn right and go straight) 2. Create an algorithm that has "if statements" and "repetition statements" that responds to user input (e.g. an object follows a mouse around the screen) 3. Translate an algorithm into a program that incorporates "if statements" and "repetition statements" that responds to user input (e.g. follow the mouse pointer around the screen) 4. Identify any errors or problems by frequently running the code at various stages of the coding process (e.g. character isn't moving when key is pressed) 5. Fix any errors and problems, and re-run program to verify fixes. (e.g. did not assign a key for that movement direction)

TDSB ICT Skills Continuum for Coding (Grades 6-8)

TDSB ICT Standards for Coding: (Grades 6-8)

- Design, develop, program and test a game (e.g., create a game in which the player guides a character through a maze using a mouse, program a character to move and jump across platforms when the arrow keys are pressed)

Grade 6	Grade 7	Grade 8
1. Identify the objectives (output) in various games (e.g., how the game is won, how a game is lost) 2. Identify what data (e.g., button) is used in various games to control the character (input) (e.g. spacebar = jump, mouse = pointer) 3. Design a game that has an objective and user-controlled input. (e.g., character moves through a maze by following the pointer of a mouse, catch falling items) 4. Write a program to move a character to an objective (e.g., complete a maze) 5. Identify any errors or problems by frequently running the code at various stages of the coding process (e.g. game does not accumulate point totals) 6. Fix any errors and problems, and re-run program to verify fixes. (e.g. forgot to add a counter c=c+1)	1. Identify how the objective (output) can be affected by the input in various ways (processing) (e.g. user controls character and game ends if a wall is hit on its way to the destination) 2. Design a game that has an objective, user-controlled input and obstacles/challenges (e.g., move a character through a maze avoiding the walls and rocks) 3. Write a program that moves a character and includes obstacles to the objective (e.g. completing an obstacle course) 4. Identify any errors or problems by frequently running the code at various stages of the coding process (e.g. character does not respond to obstacles) 5. Fix any errors and problems, and re-run program to verify fixes. (e.g. did not include the obstacle to keep checking for collision, using a loop)	1. Identify variables that have been included in a game (e.g., score, speed, name, date of birth) 2. Design a game that has an objective, user-controlled input, obstacles/challenges and variables (e.g. a character moves through a maze while collecting coins, quiz) 3. Write a program that moves a character and includes obstacles and variables to the objective and variables (e.g. completing an obstacle course with a particular score) 4. Identify any errors or problems by frequently running the code at various stages of the coding process (e.g. character responds to one obstacle but not any others) 5. Fix any errors and problems, and re-run program to verify fixes. (e.g. need to include multiple and separate code to check for each obstacle)