# Teaching Computational Thinking to Students with Learning Differences

*Emmett Wald, Westfield State University*
*CSTA New England Regional Conference*
*UMass Amherst – 12 November 2022*

Computational thinking (CT) is at the core of programming. CT is challenging for all students, but especially for those with learning differences. Considering the cognitive demands associated with various computational thinking tasks (at whatever age/grade level), alongside students' particular strengths and weaknesses, allows us to make targeted adjustments to our pedagogy that will benefit all students.

**Learning Objective**

Consider pedagogy and access from a cognitive perspective, and gain tools for teaching students with various learning styles, including those with learning differences.

**Presenter Bio**

Emmett (they/them) is an M.Ed. student and professional tutor at Westfield State University. They are passionate about equity, accessibility, and critical pedagogy in math, CS, and other STEM subjects.

Website: digitsandbytes.edublogs.org. Email: emmettmwald@gmail.com.

## Computational Thinking

Computational thinking (CT) is an essential skillset in computer science, including skills of **abstraction**, **decomposition**, **pattern recognition**, **algorithmic thinking**, and **debugging**. While there are distinct pieces to each of these skills, they are all fundamentally about "zooming in" or "zooming out" on a problem, looking at different levels of detail, and focusing on what's important while setting aside the rest.

Many skilled programmers and other CS specialists use these skills constantly, without necessarily recognizing exactly what they're doing. But in order to teach our students equitably, it's helpful for us to have a framework for understanding these more abstract (ahem) learning goals so that we can assess where students are achieving and where they are struggling and then use targeted strategies for those points of difficulty.

## CATTELL-HORN-CARROLL THEORY

The Cattell-Horn-Carroll (CHC) theory of cognitive abilities is the most **empirically-supported theory of the structure of cognitive abilities** (Flanagan & Dixon). It was developed primarily for the purpose of assessment, i.e., determining students' cognitive strengths and weaknesses. However, the taxonomy offered by the CHC theory provides a useful framework for considering the cognitive demands of certain learning tasks, as well as the intersection of these cognitive demands with our students' strengths and weaknesses.

CHC theory organizes 81 "narrow" cognitive abilities into 16 "broad" categories. The table on pages 3–4 outlines the seven broad cognitive abilities most closely associated with computational thinking, with the addition of executive functions. Executive function is not a CHC cognitive ability, but challenges with executive functioning are common and highly impactful in the classroom.

## LEARNING DIFFERENCES

"Learning differences" is an umbrella term used here to encompass students with specific **learning disabilities** as well as those with **mild intellectual disabilities**, **specific language impairments**, and/or **ADHD**. These are all students you would expect to see in a mainstream/inclusive classroom; many, though not all, will have IEPs that specify particular accommodations these students should receive. While students' IEPs can offer some useful insight and guidance for instructing them, they may fail to address CS, since it's often considered an elective or bonus subject. You may also have students who have difficulties that have not been diagnosed or that do not meet diagnostic criteria. Getting to know how students engage and perform in your class will provide an important supplement to IEPs.

While a brief definition cannot possibly capture the complexity of individual students, it is helpful to have a rudimentary sense of what specific diagnoses mean.

- Learning disabilities: neurologically-based processing disorders
  - **Auditory processing disorder**: difficulty parsing speech sounds or "filtering" sounds; may mis-hear words, especially in noisy environments
  - **Dyscalculia**: a variety of math-related disorders that may impact calculations, quantitative reasoning or abstraction, sequencing, estimating or comparing quantities, comprehension of numbers or symbols, memorization of arithmetic facts, spatial skills, or time or money math
  - **Dysgraphia**: difficulty with writing; may be motor-related, language-related, or both
  - **Dyslexia**: a variety of disorders related to word decoding; often results in reduced reading speed and comprehension
  - **Nonverbal learning disorder**: difficulty with motor control, visual–spatial abilities, social skills, and/or sensory sensitivity; often struggle with abstraction
- **Mild intellectual disability**: impairments in reasoning, problem solving, judgment, abstraction, learning, and social–emotional skills; "mild" allows for fairly independent functioning
- **Speech & language impairment**: disorders related to phonology (word sounds), syntax (sentence construction, semantics (word meanings), or pragmatics (social language)
- **Attention-deficit hyperactivity disorder**: difficulties with executive function (self-regulation, planning, organizing, decision-making, self-awareness), emotional regulation, and impulse control

# CHC Cognitive Abilities and Learning Differences in Computer Science

| Broad Cognitive Ability | Narrow Cognitive Abilities | Description | Examples in CS | Potential Challenges in CS | Associated Learning Differences |
|---|---|---|---|---|---|
| Fluid Intelligence | *Induction, sequential reasoning, quantitative reasoning* | Determining the underlying concept/process in a problem; noticing patterns and relationships; starting with defined rules/conditions and taking steps to find a solution | Developing an idea for a program that achieves a certain goal | *Sequencing, cause and effect, pattern recognition, formal logic, understanding and using algorithmic structures, complex reasoning, problem-solving, big-picture thinking* | Dyscalculia, dyslexia, intellectual disability, nonverbal learning disorder |
| Short-term Memory | *Memory span, working memory* | Remembering the order of steps or ideas after they are presented; temporarily storing and manipulating information | Copying and modifying syntax from a reference sheet | *Following directions, remembering the order of steps in a solution, holding an algorithm in memory while translating it to code, keeping track of variables and functions* | ADHD, auditory processing disorder |
| Long-term Storage & Retrieval | *Meaningful memory, associational fluency, alternative solution fluency, originality & creativity, ideational fluency* | Creative use of stored knowledge; generating relevant ideas, responses, or solutions | Adapting a previously-learned structure (e.g., loops) to use in a new | *Learning new concepts and terminology, applying existing skills in the novel context of CS, applying newly learned concepts to problems/tasks, generating possible solutions, synthesizing multiple* | ADHD, intellectual disability, speech & language impairment |
| Executive Function* | *Decision making, planning & organizing, task initiation & completion, self-monitoring, coping with frustration, metacognition (awareness of one's own thought processes)* | | Persisting in solving a sticky bug | *Getting started, staying on-task, having realistic self-expectations and goals, debugging* | ADHD, intellectual disability |

**Not a CHC cognitive ability.*

*ctd. on next page*

| **Broad Cognitive Ability** | **Narrow Cognitive Abilities** | **Description** | **Examples in CS** | **Potential Challenges in CS** | **Associated Learning Differences** |
|---|---|---|---|---|---|
| Crystallized Intelligence | *General information, language development, vocabulary, listening & communication ability* | Using general (cultural) knowledge; fluency with language; communicating ideas | Learning and using CS terms; formulating questions | *Sensory perception, sustained focus/concentration, debugging* | ADHD, auditory processing disorder, dyscalculia, dyslexia, intellectual disability, nonverbal learning disorder, speech & language impairment |
| Reading & Writing | *Reading decoding & comprehension, writing ability, etc.* | Facility with reading and writing, understanding written information, putting thoughts into writing | Interpreting written instructions for a task | *Understanding tasks or instructions, communicating with classmates, expressing confusion or asking for help, formulating an algorithm or explanation, learning technical vocabulary* | Dysgraphia, dyslexia, intellectual disability, speech & language impairment |
| Processing Speed | *Perceptual speed, rate of test taking, reading & writing fluency* | Fluency in repetitive tasks; attention, efficiency, and accuracy in simple tasks | Using correct syntax in a long list of strings | *Understanding written information or instructions, notetaking, comprehending and writing code, interpreting error messages* | ADHD, dysgraphia, dyslexia, intellectual disability, speech & language impairment |
| Decision/ Reaction Time or Speed | *Semantic processing speed, mental comparison speed, inspection* | Mental manipulation and recognition of details and differences between items | Identifying typos and syntax errors | *Comprehending code, debugging* | ADHD, dyscalculia, intellectual disability, nonverbal learning disorder |

*Adapted from Wald (2021), Table 3.*

# Instructional Strategies

Students work best when they're challenged, but not to the point of overwhelm—a state sometimes called **productive struggle**. Once we've identified the cognitive demands of a task and considered the cognitive profiles of our students, we can figure out where a certain task might push a certain student past the point of productive struggle, as well as where a student may be bored and require more stimulation. Creating built-in space for differentiation involves having supports and extensions at hand for those who need them, but also designing the core activity to be flexible in meeting students' needs.

## Universal Design for Learning

Universal Design for Learning (UDL) is a framework that emphasizes diversity of teaching material in order to reach diverse thinkers (Cast, 2018). Using multiple strategies for engaging students, presenting information, and assessing learning greatly improves the likelihood that each student will find something that works for them. UDL emphasizes a strengths-based approach rather than a deficit-based approach. Using UDL, we employ multiple avenues for:

- **Engagement**: getting students interested, helping students sustain effort and persist through difficulty, and encouraging self-regulation
- **Representation**: how students absorb information through the senses, language and symbols, and processing/comprehension
- **Action & Expression**: how students demonstrate their learning through physical action, expression and communication, and recruiting executive functions

UDL can be applied to every aspect of the teaching–learning experience. For example, when giving students instructions, you might: i) explain out loud, ii) write on the board, iii) provide a handout, and iv) demonstrate confusing steps. It's possible that you already do some or many of these things in your classroom.

In an ideal world, a perfectly-designed lesson might be truly "universal" in its application. However, it's impossible to create a truly universal design for learning. As such, we can create scaffolding in targeted locations based on what we know about our students.

## Supports for Specific Cognitive Abilities

*Adapted from Wald (2021), Appendix A.*

**Memory supports** (short-term memory, long-term storage and retrieval)

- Vocabulary: define new terms, review terms, **class glossary**, use terms frequently and **consistently**
- List or **highlight key points** during instruction
- Maintain a **reference sheet** of code/syntax
- Have students keep a list of variables and functions when coding
- Use **anchor charts** for important facts, procedures, etc.
- **Review** and re-teach as needed

**Communication supports** (crystallized intelligence, reading/writing)

- UDL: provide information in **multiple formats**; give students multiple means of expression
- Class notes/handouts with key points
- **Frameworks** for interpreting incoming info, **scripts** for communicating with others: how to make sense of an error message; how to ask for help
- Providing **prompts and cues** if students are struggling to express themselves

**Processing** (processing speed, decision/reaction time) & **executive function supports**

- Allow **more time** for processing and responses (offer bonus exercises or free exploration time to those who work quickly)
- Monitor student progress, build **checkpoints** into activities
- Physical manipulatives and **kinesthetic** activities
- Allow students to take **breaks**; offer a **quiet area** to work or take breaks
- Scaffold difficult tasks
  - **Examples, templates, or outlines** for the planning/organizing stage
  - Make sense of the problem and write an algorithm/outline before beginning to code
  - Questions to **guide the debugging** process
  - A set of options to try when something isn't working
- Maintain a **balance** with open inquiry: don't over-support, allow **productive struggle**, ask guiding questions

**Fluid intelligence supports**

- Explicit instruction
  - Prioritize big ideas; set clear goals and expectations
  - **Review** prior learning
  - Use **modeling** and step-by-step demos
  - Monitor student performance and provide **immediate corrective feedback**
- Modeling with UDL: demos with notes and **sample code**, video **tutorials**, exploring or modifying sample code, "**unplugged**" demonstrations
- Use **flowcharts** for algorithm design
- Levels of abstraction (problem, algorithm, program, and execution)*
  - Be clear and explicit; use language to **differentiate** between levels
  - **Anchor chart** diagram of different levels
  - Emphasize **higher levels** of abstraction
- Reflection: comprehension checkpoints, class discussion, **formative assessments**

**For more information about levels of abstraction, see Armoni (2013) or Wald (2021).*

## References & Further Reading

Wald, E. (2021). "Instructional Strategies for Teaching Computational Thinking to K-12 Students with Learning Differences." https://digitsandbytes.edublogs.org/files/2022/10/Instructional-Strategies-CS-for-LDs.pdf

**Computational Thinking**

Armoni, M. (2013). "On Teaching Abstraction in Computer Science to Novices." *Journal of Computers in Mathematics and Science Teaching*, *32*(3), 265–284.

Perrenet, J., & Kaasenbrood, E. (2006). *Levels of Abstraction in Students' Understanding of the Concept of Algorithm: The Qualitative Perspective*. *3(2)*, 270–274. https://doi.org/10.1145/1140124.1140196

Valenzuela, J. (2020). "How to develop computational thinkers." *International Society for Technology in Education*. https://www.iste.org/explore/how-develop-computational-thinkers

**Cattell-Horn-Carroll theory**

Flanagan, D. P., & Dixon, S. G. (2014). "The Cattell-Horn-Carroll Theory of Cognitive Abilities." *Encyclopedia of Special Education*. John Wiley & Sons. https://onlinelibrary.wiley.com/doi/full/10.1002/9781118660584.ese0431

**Learning Differences**

Gage, N. A., et al. (2012). "Characteristics of Students With High-Incidence Disabilities Broadly Defined." *Journal of Disability Policy Studies*, *23*(3), 168–178. https://doi.org/10.1177/1044207311425385

Learning Disabilities Association of America (2013). *The Ins and Outs of Learning Disabilities*. https://ldaamerica.org/info/the-ins-and-outs-of-learning-with-ld/

*Characteristics of Children with Learning Disabilities*. National Association of Special Education Teachers. https://www.naset.org/fileadmin/user_upload/LD_Report/Issue__3_LD_Report_Characteristic_of_LD.pdf

**Instructional Strategies**

CAST (2018). *Universal Design for Learning Guidelines version 2.2*. https://udlguidelines.cast.org/

Hansen, A. K., et al. (2016). *Differentiating for Diversity: Using Universal Design for Learning in Computer Science Education*. SIGCSE, Memphis, TN. https://doi.org/10.1145/2839509.2844570

Outlier Research & Evaluation. Teaching Practices Guide: Improving Accessibility for Students with Learning Disabilities & ADHD: The Computer Science Principles (CSP) Course. outlier.uchicago.edu/accessCSP/