Integrating CT Working Session

Agenda

- 10 min. Discuss integrations you have done or seen
- 20 min. Develop example(s) of integrations you'd like to see or have done.
- 10 min. Report out and sign up to continue work….

Roles

During discussion phase: **Facilitator** starts a structured discussion by asking group members to share out an idea, resources for a lesson that integrates CT into another subject in K8. **Scribe** takes notes on sticky notes or online.

During development phase: **Facilitator** asks for ideas for a new CT examples in the content area then helps to select and develop a new or modified example (increasing the CT). **Scribe** takes notes on sticky notes or online.

During reporting out phase: **Facilitator and/or scribe** reports out on example their group developed.

CT concept or capability	Definition	What to do	Why is this important?
Data Collection	The process of gathering appropriate information	When developing solutions that a computer can execute, determine what data is needed and appropriate to use in the problem solving process.	It is necessary to use relevant data in the problem solving process. There is a saying "Garbage in, garbage out".
Data Analysis	Making sense of data, finding patterns, and drawing conclusions	When developing solutions that a computer can execute, analyze output data to understand and improve the process applied to the input data.	it is necessary to analyze output data to assess the outcome of a process and/or correctness of a solution.
Data Representation	Depicting and organizing data in appropriate graphs, charts, words, or images	When coming up with solutions that computers can execute, plan ahead for data input and output, data storage, and easy data retrieval. Come up with compact representations. Develop a plan for visualizing data sets.	Easy storage and retrieval of data will make your code faster and memory usage more efficient.
Problem Decomposition	Breaking down tasks into smaller, manageable parts	When coming up with solutions that computers can execute, break long sequences of code down into smaller parts to write more compact, reuseable code.	Smaller reuseable chunks of code (subprocedures) are easier to debug, test, and reuse. Sometimes breaking things down also enables us to attack problems as independent smaller problems.
Abstraction	Reducing complexity to define main idea	When coming up with solutions that computers can execute, develop simple representations that can stand for the set of all such parts. Reduce detail and complexity to define the main idea. Seek regularity so one can stand for all.	To achieve scale and wide applicability of solutions develop general case solutions not solutions to one instance of a problem.
Algorithms & Procedures	Series of ordered steps taken to solve a problem or achieve some end.	When coming up with solutions that computers can execute, consider the capabilities of the computer and the language specific commands it can follow. Develop a solution using a sequence of instructions that the computer can execute.	Computers only follow instructions they are given in a language they can understand.
Automation	Having computers or machines do repetitive or tedious tasks.	When coming up with solutions that computers can execute, describe how a machine or computer could be used to complete repetitive tasks.	Computers can easily perform repetitions - thus they can easily repeat a task, search a long list, produce all possible combinations, and/or do random selection.
Parallelization	Organize resources to simultaneously carry out tasks to reach a common goal.	Develop a plan on how the computer can be used to carry out more than one process at the same time.	Computers (those with multiple processors) can readily perform multiple processes at once but when doing so, it is important to make sure tasks do not interfere with one another.
Simulation	Dynamic representation or model of a process. Simulation also involves running experiments using models.	When studying and potentially solving real-world problems, develop a dynamic computer model of a real-world scenario within which to run experiments.	Computers can be used to run experiments that take place in virtual worlds. This is specially important when the scenario or process being studied is too fast, too slow, too expensive, or too dangerous to experiment with in the real-world.

Operational Definition of Computational Thinking

for K–12 Education

The International Society for Technology in Education (ISTE) and the Computer Science Teachers Association (CSTA) have collaborated with leaders from higher education, industry, and K–12 education to develop an operational definition of computational thinking. The operational definition provides a framework and vocabulary for computational thinking that will resonate with all K–12 educators. ISTE and CSTA gathered feedback by survey from nearly 700 computer science teachers, researchers, and practitioners who indicated overwhelming support for the operational definition.

Computational thinking (CT) is a problem-solving process that includes (but is not limited to) the following characteristics:

- Formulating problems in a way that enables us to use a computer and other tools to help solve them.
- Logically organizing and analyzing data
- Representing data through abstractions such as models and simulations
- Automating solutions through algorithmic thinking (a series of ordered steps)
- Identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources
- Generalizing and transferring this problem solving process to a wide variety of problems

These skills are supported and enhanced by a number of dispositions or attitudes that are essential dimensions of CT. These dispositions or attitudes include:

- Confidence in dealing with complexity
- Persistence in working with difficult problems
- Tolerance for ambiguity
- The ability to deal with open ended problems
- The ability to communicate and work with others to achieve a common goal or solution

 This material is based upon work supported by the National Science Foundation under Grant No. CNS-1030054.