Exploring concepts in Computational Thinking

Sonit Singh

[With materials from Sarah Boyd, Mark Dras and Diego Molla-Aliod]
Learning Objectives

- Where references to algorithms feature in the Australian Curriculum: Digital Technologies
- The definition and key characteristics of Computational Thinking
- How to explain an algorithm
- Plugged and unplugged activities for teaching algorithms in the primary context
- About visual programming environments available
New Curriculum (Starting from 2019)

- Science (incorporating K–6 Science and Technology) K–10 (Current)
- Science and Technology K–6 (NEW from 2019)
- Spanish K–10 (Current)
- Spanish K–10 (NEW from 2019)

- Technology (Mandatory) 7–8 (Current)
- Technology Mandatory Years 7–8 (NEW from 2019)
- Textiles Technology 7–10
Where to find Syllabus?

Syllabus

The Science and Technology K–6 Draft Syllabus Consultation Report (PDF, 604KB, 42 pages) and new Science and Technology K–6 Syllabus were developed using the NESA syllabus development process.
Foundation- 2: students follow, describe and represent a sequence of steps and decisions (algorithms) to solve problems (ACTDIP004) and this includes the application of abstraction, which involves highlighting the most important information and steps.

Years 3- 4: students build on the previous learning objective by beginning to define simple problems and include branching (decisions) and user input in their design (ACTDIP011).

Year 3 onwards: students have opportunities to implement their algorithms with visual programs.

Years 5 - 6: students integrate repetition into their visual programs (ACTDIP020) and begin to develop more comprehensive designs, as well as modify and follow designs represented diagrammatically and in English (ACTDIP019).
What are Digital Technologies & Computational Thinking

https://www.youtube.com/watch?v=Uqyu6iJ9h_o
Why learn Digital Technologies?

Digital Technology, is not just about sitting in a room playing with a computer - it is so much more than that. Digital Technology is about changing the world.

The study of digital technology and computational thinking into Australian classrooms ensures that Australian youth move from being users of technology, produced elsewhere, to becoming world leaders in developing new technological innovations and solutions.

Digital technology will become more pervasive as our society evolves and it will be essential for our next generations to understand how technology works, in order to have the best insight into how technology can benefit society and the environment. It's also about understanding how software works so that, in any industry, we can consider the best ways to integrate technology to have the greatest impact.
This video highlights Australia’s great inventions. Many more of our future inventions will increasingly involve computing and code.

Source: https://www.youtube.com/watch?v=THEpcW7vFkc
What is Computational Thinking

High-level view

**Decomposition**
Breaking the problem into smaller, more manageable parts.

**Pattern recognition**
Recognising which parts are the same and the various attributes we can use to define them.

**Abstraction**
Filtering out the data you need and what you don’t based on the attributes.

**Algorithm design**
Planning the step-by-step instructions that need to be carried out to achieve the goal.

Source: Jo Culf, CSER MOOC
What is Computational Thinking

Deeper View

Concepts
- Logic: predicting & analysing
- Algorithms: making steps & rules
- Decomposition: breaking down into parts
- Patterns: spotting & using similarities
- Abstraction: removing unnecessary detail
- Evaluation: making judgement

Approaches
- Tinkering: experimenting & playing
- Creating: designing & making
- Debugging: finding & fixing errors
- Persevering: keeping going
- Collaborating: working together

The Computational Thinker: Concepts & Approaches

www.barefootcas.org.uk

Barefoot would like to acknowledge the work of Julia Briggs and the eLMI team at Somerset County Council for their contribution to this poster.
Logical reasoning helps us explain why something happens.

Computers don’t make things up as they go along or work differently depending on how they happen to be feeling at the time. This means that they are predictable. Because of this we can use logical reasoning to work out exactly what a program or computer system will do.
Logical Reasoning…

https://barefootcas.org.uk/barefoot-primary-computing-resources/concepts/logic/

• In English, pupils might explain what they think a character will do next in a novel, or explain the character’s actions in the story so far.
• In science, pupils should explain how they have arrived at their conclusions from the results of their experiments.
• In history, pupils should discuss the logical connections between cause and effect; they should understand how historical knowledge is constructed from a variety of sources.
• In design and technology, pupils reason what material is best suited to each part of their projects.
• In philosophy for children sessions, pupils can use logical reasoning to analyse arguments.
Logical Reasoning: Activity

https://barefootcas.org.uk/barefoot-primary-computing-resources/concepts/logic/

World Map Logic Activity

Home → My Curriculum – England → Use logical reasoning to predict the behaviour of simple programs → World Map Logic Activity

World map logic: An introduction to logic

Recommended Age Group: 5 – 7 years
Activity Duration: 30 mins
An algorithm is a sequence of instructions or a set of rules to get something done.

You probably know the fastest route from school to home, for example, turn left, drive for five miles, turn right. You can think of this as an ‘algorithm’ – as a sequence of instructions to get you to your chosen destination. There are plenty of algorithms (i.e. routes) that will accomplish the same goal; in this case, there are even algorithms (such as in your satnav) for working out the shortest or fastest route.

Algorithms are written for a human, rather than for a computer to understand. In this way algorithms differ from programs.
What do Algorithms look like?

https://barefootcas.org.uk/sample-resources/algorithms/

- A lesson plan can be regarded as an algorithm for teaching a lesson.
- There will be a sequence of steps pupils follow for many activities, such as getting ready for lunch or going to PE.
- In cookery, we can think of a recipe as an algorithm.
- In English, we can think of instructional writing as a form of algorithm.
- In science, we might talk about the method of an experiment as an algorithm.
- In maths, your approach to mental arithmetic (or many computer-based educational games) might be an implementation of a simple algorithm.
Algorithms: Activities

https://barefootcas.org.uk/sample-resources/algorithms/

How to draw a crazy character algorithm

1. Draw a circle for the body
2. Add 2 eyes
3. Add a crown
4. Add wings
5. Add four legs

Example of a completed worksheet
The process of breaking down a problem into smaller manageable parts is known as decomposition. Decomposition helps us solve complex problems and manage large projects.

The problem of making breakfast can be decomposed into a number of tasks:

- Make breakfast
  - Make toast
    - Slice bread
    - Toast bread
    - Butter toast
    - Add jam
  - Make tea
    - Boil water
    - Brew tea
    - Add milk

Two people could make this breakfast at the same time, one could make tea and one could make toast.

A labelled diagram of a flowering plant. We find out more as we decompose.
Decomposing problems into their smaller parts is not unique to computing: it’s pretty standard in engineering, design and project management.

Software development is a complex process, and so being able to break down a large project into its component parts is essential – think of all the different elements that need to be combined to produce a program, like PowerPoint.

The same is true of computer hardware: a smartphone or a laptop computer is itself composed of many components, often produced independently by specialist manufacturers and assembled to make the finished product, each under the control of the operating system and applications.

Why is Decomposition important?

https://barefootcas.org.uk/sample-resources/decomposition/
Decomposition: Activity

Simple labeling in early years.

Sequencing in early years.
Patterns

https://barefootcas.org.uk/barefoot-primary-computing-resources/concepts/patterns/

- Patterns are everywhere, for example, we use weather patterns to create weather forecasts; children might notice patterns in how teachers react to their behaviour to work out how to behave next time.
- By identifying patterns we can make predictions, create rules and solve more general problems.
- In computing, the method of looking for a general approach to a class of problems is called generalisation.
What do Patterns look like?

https://barefootcas.org.uk/barefoot-primary-computing-resources/concepts/patterns/

- From an early age, they’ll become familiar with repeated phrases in nursery rhymes and stories; later on they’ll notice repeated narrative structures in traditional tales or other genres.
- In music, children will learn to recognise repeating melodies or bass lines in many musical forms.
- In English, pupils might notice common rules for spellings, and their exceptions.
- In maths, pupils typically undertake investigations in which they spot patterns and deduce generalised results.

Can pupils spot the pattern to reveal the number sequence rule?

16, 35, 73, 149 ____

Possible answers: 303, 302, 304, 301
**Patterns: Activity**

https://barefootcas.org.uk/barefoot-primary-computing-resources/concepts/patterns/
Abstraction

https://barefootcas.org.uk/barefoot-primary-computing-resources/concepts/abstraction/

• Abstraction is about simplifying things; identifying what is important without worrying too much about the detail. Abstraction allows us to manage complexity.

• We use abstractions to manage the complexity of life in schools. For example, the school timetable is an abstraction of what happens in a typical week: it captures key information such as who is taught what subject where and by whom, but leaves to one side further layers of complexity, such as the learning objectives and activities planned in any individual lesson.

![School Timetable Example](image-url)
Why is Abstraction important?

https://barefootcas.org.uk/barefoot-primary-computing-resources/concepts/abstraction/

• Abstractions are sometimes represented as layers or hierarchies, allowing us to view things at different degrees of detail. The nature of being able to hide complexity within boxes makes abstraction a powerful tool as we do not need to worry about the technical detail of what goes on inside each box.

• In computer science, abstraction is used to manage the complexity of much of what is designed and created. Computer hardware is seen as components or black boxes. Software is built of layers each hiding the complexity of the next successive layer.
A database of country data only holds selected data, such as population, area. Image by kind permission of Simon Haughton.
Abstraction: Activity


Modelling the internet activity: Understanding how the internet provides services such as the world wide web

Pupils use slips of paper to represent the packets of data (red dots) moving around the internet. This is a very simplified abstraction that helps them start to learn about packets, servers and IP addresses.
Evaluation

• Evaluation is about making judgements, in an objective and systematic way where possible.
• Evaluation is something we do every day – we make judgements about what to do and what we think based on a range of factors.
• For example, when considering a new digital device for use in the classroom, there would be a number of criteria that would be considered; for example, operating system, portability, memory, screen size, ease of use and warranty.
Evaluation is something that occurs everyday in schools; pupils evaluate their work, teachers evaluate lessons that they deliver and pupils’ learning and progress is evaluated.
What everyday algorithms can you think of?
Algorithms in our daily life

Source: code.org  https://www.youtube.com/watch?v=FHzuEh1kJ18
Algorithms in our daily life

Source: code.org  https://www.youtube.com/watch?v=AWqo8Gxtrjs
An algorithm is a **step-by-step description** of how to solve a problem. It involves identifying **what information is needed** to solve the problem and **how that information is used**.

**Computational thinking** (sequencing, decomposition and abstraction) is used to design and create effective solutions.

Computers require **explicit instructions**. Humans tell computers what to do with **programming languages**.
Activity: Unplugged Pencil-Coders!

Access: pencilcode.net or blockly-games.appspot.com/turtle

1. Find a partner/work in small groups.
2. One person scribes.
3. Think of a simple shape, letter or number.
4. Decide on your language.
5. Write your instructions.
6. Swap instructions and try to follow each other's' algorithms.

Extension: Incorporate Pencil Code or Blockly Turtle
Flowcharting

Flowcharts are used by programmers to **plan** their algorithms.

A great way to **plan, test and debug ideas** without committing lots of time or effort.

Can be used to **guide coding**.
Conditions (IF/ELSE) [Decisions]

Source: code.org

https://www.youtube.com/watch?v=JtL7w6ja5il
Loops (Repetition/Iterations)

Source: code.org

https://www.youtube.com/watch?v=d7e48cYq7uc
Activity: Algorithm Flowchart

1. Think of an everyday activity or situation where you need to make a decision.
3. Create a flowchart that represents your idea as an algorithm.
4. Share your algorithm!
Activity: Algorithms in Visual Programming

Blockly Games:

blockly-games.appspot.com

Select “Maze” or “Turtle”
Activities

https://barefootcas.org.uk/activities/

Crazy Character Algorithms Activity: An introduction to sequences of instructions

**Recommended Age Group:** 5 – 7 years  
**Activity Duration:** 30 minutes

### Concepts and approaches

- **Algorithms**
- **Decomposition**
- **Logic**
- **Debugging**

![How to draw a crazy character algorithm]

- draw a circle for the body
- add 2 eyes
- add a crown
- add wings
- add four legs

An example crazy character algorithm
Activities

https://barefootcas.org.uk/safety-snakes-download/

What pupils will learn

Safety Snakes provides an excellent way for young pupils to learn about good and unwise online behaviours. As they make their way around the game board, they will land on squares providing details of different types of online scenarios. Good scenarios are placed at the bottom of ladders, so pupils are rewarded with a lift onto a higher square when they land there. Squares with details of unwise scenarios sit at the top of the snakes, so pupils learn that they literally go backwards if they land on these squares. This ‘carrot and stick’ methodology helps pupils understand what they should and should not do when they are using the internet.

What does the resource pack contain?

The Safety Snakes resource pack contains:

- The Safety Snakes Bee-Bot mat
- A fake-bot (so the game can be played without a Bee-Bot)
- A set of instructions
- A Scratch online version of the game
- Teacher notes
- A PowerPoint presentation of the game
More Activities

https://barefootcas.org.uk/activities/sen/

Cake Shopping List Activity: Developing Abstraction Skills

River Crossing Activity: Developing Logical Reasoning

Getting Ready for School Decomposition

Lego Building Algorithm Activity
# Development Process: Game Creation

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Defining and understanding the problem</strong></td>
<td>What is the broad topic/problem? What knowledge is required? Goal of the game? What is needed to create the game?</td>
</tr>
<tr>
<td><strong>Designing and Planning the solution</strong></td>
<td>Storyboard, flowchart, mind-map, mock-design, sketch, user interface design...</td>
</tr>
<tr>
<td><strong>Implementing the solution</strong></td>
<td>Create it!</td>
</tr>
<tr>
<td></td>
<td>- crafts</td>
</tr>
<tr>
<td></td>
<td>- programming environment (e.g. Scratch)</td>
</tr>
<tr>
<td><strong>Testing &amp; Evaluating solution</strong></td>
<td>(Iterative) Test game, find “bugs” and make improvements! Does it achieve the goal? Usability? Fun?</td>
</tr>
<tr>
<td><strong>Communicating ideas</strong></td>
<td>Share the game (online, pitch idea, have friends/family or peers play)</td>
</tr>
</tbody>
</table>
Think of a community problem and design a game to educate others. For example being prepared for a cyclone or flood.
Exploring Game Design Environments

http://www.scratchjr.org
https://scratch.mit.edu/
http://scratched.gse.harvard.edu/ (teacher resources)
http://www.cs-first.com/course/game-design
Game Development in Scratchjr
Year 2 example (shared by Jim Cash) on Youtube
https://www.youtube.com/watch?time_continue=1&v=SbGuN_Ety_g
MaKey MaKey – An invention kit for everyone

https://www.youtube.com/watch?time_continue=118&v=rfQqh7iCcOU
An Inspirational video:

Look Inside.™: Mick Ebeling | Intel

https://www.youtube.com/watch?v=ol19tt3VWhQ
• What did you get out of today?
• How are you going to use what we did today in the classroom?
• What did you like about today?
• What would you have changed?
Putting it into the Classroom

"Flesh out a lesson appropriate for your stage to teach one of the concepts you have learnt about today; e.g. abstraction, algorithms, patterns, decomposition."

Try and collect everyone's ideas and circulate them to each other
Recommended Resources

- Code.org
- CS Unplugged
- Blockly Games
- csermoocs.adelaide.edu.au