

Digital flashcards

**Lessons:** 5

**Programming languages:** MakeCode

**Target age:** 7-11 yrs

**Subjects & topics:**

* Computational thinking: Algorithms, Abstraction, Pattern recognition
* Programming: Sequence
* Languages: Vocabulary

# Unit of work summary

This series of five lessons is aimed at students aged 7-8 years and builds on the [‘Nature art’](https://microbit.org/teach/lessons/nature-art-unit-of-work/) unit. Students design sequenced algorithms for flashcards to help them learn a foreign language, developing their understanding of computational thinking. They then write programs to create digital flashcards using the micro:bit and test and evaluate their work.

## Overall key learning

* can understand and apply the fundamental principles and concepts of computer science, including abstraction, logic, algorithms and data representation
* can analyse problems in computational terms, and have repeated practical experience of writing computer programs in order to solve such problems
* can evaluate and apply information technology, including new or unfamiliar technologies, analytically to solve problems
* are responsible, competent, confident and creative users of information and communication technology

## Additional skills

Creative thinking, collaboration, problem-solving

## Lesson 1: Flashcard algorithms

In this ‘unplugged’ lesson pupils use flashcards to practice words learnt in another language. They consider the ‘responder’ and ‘shower’ roles of the people using the flashcards and write, test and debug algorithms for others to follow.

**Key learning:**

* To know and understand what algorithms are
* To write algorithms with clear instructions
* To test and debug algorithms

## Lesson 2: Abstraction & programming

Pupils recap their learning from the [Nature art](https://microbit.org/teach/lessons/nature-art-unit-of-work/) unit before being introduced to using the BBC micro:bit as a digital flashcard. They choose language vocabulary then write sequenced programs using the MakeCode editor to displays LED images of the words.

**Key learning:**

* To use abstraction when planning LED images
* To write programs that create LED images
* To sequence programs

## Lesson 3: Patterns & delays

Pupils develop their understanding of the ‘wait’ command, using it in algorithms and micro:bit MakeCode programs and identifying patterns.

**Key learning:**

* To identify solutions to problems
* To identify patterns
* To use delays in algorithms and programs

## Lesson 4: Predicting & experimenting

Pupils develop their logical reasoning skills before experimenting with the MakeCode editor to find additional ways of controlling the BBC micro:bit’s LEDs. They then plan an algorithm for a digital number flashcard.

**Key learning:**

* To use logical reasoning to identify the output of a program
* To tinker (experiment) to develop understanding
* To create an algorithm that meets given criteria

## Lesson 5: Debugging & evaluating

Pupils program the BBC micro:bit as a digital number flashcard and evaluate their programs against the design criteria before reviewing their learning from this unit.

**Key learning:**

* To follow an algorithm accurately to create a digital number flashcard
* To write and debug programs that meets design criteria
* To evaluate against design criteria

# Curriculum links

These lessons are mapped to the following learning objectives and standards for computing and foreign languages:

## England National Curriculum

#### KS2 computing curriculum

Curriculum aims:

* can understand and apply the fundamental principles and concepts of computer science, including abstraction, logic, algorithms and data representation
* can analyse problems in computational terms, and have repeated practical experience of writing computer programs in order to solve such problems
* can evaluate and apply information technology, including new or unfamiliar technologies, analytically to solve problems
* are responsible, competent, confident and creative users of information and communication technology

Students should be taught to:

* design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts
* use sequence, selection, and repetition in programs; work with variables and various forms of input and output
* use logical reasoning to explain how some simple algorithms work and to detect and correct errors in algorithms and programs

[Read the full KS2 computing curriculum.](https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/239033/PRIMARY_national_curriculum_-_Computing.pdf)

#### KS2 languages curriculum

Students should be taught to:

* explore the patterns and sounds of language through songs and rhymes and link the spelling, sound and meaning of words
* develop accurate pronunciation and intonation so that others understand when they are reading aloud or using familiar words and phrases

[Read the full KS2 languages curriculum.](https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/239042/PRIMARY_national_curriculum_-_Languages.pdf)

#### KS2 DT curriculum

Students should be taught to:

* apply their understanding of computing to program, monitor and control their products

[Read the full KS2 DT curriculum](https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/239041/PRIMARY_national_curriculum_-_Design_and_technology.pdf)

## Scotland Curriculum for Excellence

#### Technologies

* I can explore and experiment with digital technologies and can use what I learn to support and enhance my learning in different contexts (TCH 1-01a)
* I can explore and comment on processes in the world around me making use of core computational thinking concepts and can organise information in a logical way (TCH 1-13a)
* I understand the instructions of a visual programming language and can predict the outcome of a program written using the language (TCH 1-14a)
* I can demonstrate a range of basic problem solving skills by building simple programs to carry out a given task, using an appropriate language (TCH 1-15a)
* I understand the operation of a process and its outcome. I can structure related items of information (TCH 2-13a)
* I can explain core programming language concepts in appropriate technical language (TCH 2-14a)
* I can create, develop and evaluate computing solutions in response to a design challenge (TCH 2-15a)

[Read the full Curriculum for Excellence: technologies](https://education.gov.scot/Documents/Technologies-es-os.pdf)

#### Modern languages

* Reading, finding and using information - I can work on my own or with others to demonstrate my understanding of words and phrases containing familiar language (MLAN 1-08b)
* Reading, finding and using information - I can read and demonstrate understanding of words, signs, phrases and simple texts containing mainly familiar language (MLAN 2-08b)

[Read the full Curriculum for Excellence: modern languages](https://www.education.gov.scot/Documents/modern-languages-eo.pdf)

## Northern Ireland Curriculum - Primary

#### Using ICT across the curriculum

Pupils should be taught to:

* explore - investigate, make predictions and solve problems through interaction with digital tools
* express - create information and multimedia products using a range of assets
* exchange - communicate using a range of contemporary methods and tools
* evaluate - talk about, review and make improvements to work, reflecting on the process and outcome

#### KS1 - suggested curriculum ideas for the world around us

* design and make simple models

#### KS2 suggested curriculum ideas for the world around us

* design and make models

[Read the full Northern Ireland curriculum - primary](https://ccea.org.uk/downloads/docs/ccea-asset/Curriculum/The%20Northern%20Ireland%20Curriculum%20-%20Primary.pdf)

#### KS1 & KS2 - requirements for using ICT

* explore - investigate, make predictions and solve problems through interaction with digital tools

[Read the full KS1 & KS2 requirements for using ICT](https://ccea.org.uk/downloads/docs/ccea-asset/Curriculum/Curriculum%20Requirements%20for%20Using%20ICT.pdf)

#### Primary using ICT - desirable features - computational thinking and coding

**Level 4**

Pupils should:

* create a more sophisticated coding project using a broad range of commands; and/or
* solve a given problem using commands in a programming environment.

**Programmable devices (such as Parrot Drone, micro:bit or Sphere)**

* look at and talk about examples of coding projects, including the use of motion, looks, lights or sounds, sensors, control and events such as ‘if...then’ and ‘loop until’ (or equivalent) that make the code more efficient;
* recognise that these projects are composed of different components and break the task into smaller manageable tasks (decomposition);
* in small groups, plan and storyboard their own coding project, working out what different parts of the program must do, using logical reasoning to discuss and compare the commands that are required for their algorithm;
* use a range of commands to create a project including triggering commands such as ‘if...then’ and ‘loop until’ to facilitate a more efficient method of interaction;
* test and debug at regular intervals and collaborate with others to solve problems as they arise;

**Finally**

* share their work (possibly using digital tools), respond to feedback and comment on others’ work; and
* organise files and export work in an appropriate format so that others may view it.

[Read all Primary using ICT desirable features](https://ccea.org.uk/downloads/docs/ccea-asset/Curriculum/Primary%20Using%20ICT%20Desirable%20Features%20Update%202019.pdf)

## Curriculum for Wales

#### Science and technology

Progression step 2 - computation is the foundation for our digital world:

* I can safely use a range of tools, materials and equipment to construct for a variety of reasons
* I can use computational thinking techniques, through unplugged or offline activities
* I can create simple algorithms and am beginning to explain errors
* I can follow algorithms to determine their purpose and predict outcomes
* I can follow instructions to build and control a physical device

Progression step 3 - computation is the foundation for our digital world:

* I can explain and debug algorithms

[Read the full science and technology curriculum](https://hwb.gov.wales/curriculum-for-wales/science-and-technology/descriptions-of-learning/)

#### Languages

Progression step 2 - languages connect us:

* I am beginning to draw on information presented in one language and convey it in my own words in another

Progression step 3 - languages connect us:

* I can receive information in one language and adapt it for various purposes in another language

Progression step 3 - understanding languages is key to understanding the world around us:

* I can recognise high-frequency words and phrases and understand the general meaning in what I hear, read and see

[Read the full languages curriculum](https://hwb.gov.wales/curriculum-for-wales/languages-literacy-and-communication/)

#### Digital competence framework

Progression step 1 - data and computational thinking - problem-solving and modelling:

* I can identify, create and follow sequences and patterns in everyday activities.
* I can recognise and follow instructions in the appropriate order to perform a task.
* I can organise, select and use simple language to give instructions to others.
* I can control devices giving instructions.
* I can identify errors in simple sets of instructions (algorithm).

Progression step 2 - data and computational thinking - problem-solving and modelling:

* I can detect and correct mistakes which cause instructions (a solution) to fail (debug).
* I can create and record verbal, written and symbolic instructions to test ideas, e.g. the order of waking up through a diagram or flowchart.
* I can change instructions to achieve a different outcome.

Progression step 3 - data and computational thinking - problem-solving and modelling:

* I can understand the importance of the order of statements within algorithms.

Progression step 1 - producing – creating digital content:

* I can create simple digital work.

Progression step 1 - producing - evaluating and improving digital content:

* I can comment on work in relation to a single success criterion.

Progression step 2 - producing - evaluating and improving digital content:

* I can give an opinion about my own work and suggest improvements based on the success criteria.

[Read the digital competence framework](https://hwb.gov.wales/curriculum-for-wales/cross-curricular-skills-frameworks/digital-competence-framework)

## USA Code.org

**CS Fundamentals**

Courses A and B

Concepts included:

* computational thinking
* algorithms & programming
* sequencing
* events
* debugging

[Read the full Code.org CS Fundamentals curriculum](https://code.org/educate/curriculum/elementary-school).

## USA CSTA Standards

#### Grades 3-5

1B-CS-01 - Describe how internal and external parts of computing devices function to form a system.

1B-CS-02 - Model how computer hardware and software work together as a system to accomplish tasks

1B-CS-03 - Determine potential solutions to solve simple hardware and software problems using common troubleshooting strategies.

1B-AP-08 - Compare and refine multiple algorithms for the same task and determine which is the most appropriate.

1B-AP-10 - Create programs that include sequences, events, loops, and conditionals.

1B-AP-11 - Decompose (break down) problems into smaller, manageable subproblems to facilitate the program development process.

1B-AP-12 - Modify, remix, or incorporate portions of an existing program into one's own work, to develop something new or add more advanced features.

1B-AP-15 - Test and debug (identify and fix errors) a program or algorithm to ensure it runs as intended.

[Read the CSTA Standards in full](https://csteachers.org/k12standards/ ).

This content is published under a [Creative Commons Attribution-ShareAlike 4.0 International (CC BY-SA 4.0)](https://creativecommons.org/licenses/by-sa/4.0/) licence.