

Computing fundamentals

**Lessons:** 6

**Programming languages:** MakeCode

**Target age:** 11-14 yrs

**Subjects & topics:**

* Computational thinking: Algorithms, Flowcharts, Pseudocode
* Programming: Selection, Iteration
* Computer systems: Hardware & software, Input/output

# Unit of work summary

This series of six lessons is aimed at students in the first year of secondary school. Students are introduced to the core concepts of computational thinking, programming and computer systems through unplugged activities and learning with the BBC micro:bit. No prior learning is assumed and this is an ideal introduction for students to ensure they have a shared understanding of these important elements of computing.

## Overall key learning

* understand and apply the fundamental principles & concepts of computer science.
* gain practical experience of writing computer programs to solve problems.
* evaluate and apply information technology, including new or unfamiliar technologies analytically to solve problems
* be responsible, competent, confident and creative users of information and communication technology.

## Additional skills

Problem-solving, collaboration, creative thinking

## Lesson 1: Computational thinking 1

Students are introduced to computational thinking through a team challenge. They create and test a giant paper aeroplane and create an algorithm for someone else to follow.

**Key learning:**

* To understand what ‘computational thinking’ is
* To develop computational thinking skills
* To write an accurate algorithm

## Lesson 2: Computational thinking 2

Students deepen and extend their learning of computational thinking by considering different applications of the concepts and designing a prototype of a computerised paper aeroplane.

**Key learning:**

* To develop a deeper understanding of computational thinking concepts
* To understand the steps in computational thinking
* To develop and present a prototype and algorithm for a computerised paper aeroplane

## Lesson 3: Programming 1

Students learn how algorithms are followed to create programs, using flowcharts and pseudocode, and the importance of debugging and testing programs.

**Key learning:**

* To understand the relationship between algorithms and programming
* To understand and use pseudocode and flowchart algorithms
* To tinker, test and debug to create a working program using a graphical programming language

## Lesson 4: Programming 2

Students experiment with iteration and selection in algorithms and programs and explore using graphical and text-based programming languages.

**Key learning:**

* To understand iteration and selection and why they are used
* To develop algorithms and programs using iteration and selection
* To experiment with graphical and text-based programming languages

## Lesson 5: Computer systems 1

Students develop their understanding of computer systems, learning about inputs and outputs, hardware and software,applying their understanding to create algorithms and write programs using the BBC micro:bit.

**Key learning:**

* To know and understand the common features of computer systems
* To be able to explain input and output devices, hardware and software
* To apply understanding to writing algorithms and programming micro:bit

## Lesson 6: Computer systems 2

Students showcase their understanding of the BBC micro:bit by creating a short explainer video including what they have learnt about computer systems, programming and computational thinking.

**Key learning:**

* To plan and create a short explainer video about micro:bit
* To follow criteria and use criteria to evaluate
* To review and evaluate learning

# Curriculum links

## England National Curriculum

#### KS3 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, use and evaluate computational abstractions that model the state and behaviourof real-world problems and physical systems
* use logical reasoning to compare the utility of alternative algorithms for the same problem
* use two or more programming languages, at least one of which is textual, to solve a variety of computational problems
* understand the hardware and software components that make up computer systems, and how they communicate with one another and with other systems
* understand how instructions are stored and executed within a computer system
* undertake creative projects that involve selecting, using, and combining multiple applications, preferably across a range of devices, to achieve challenging goals, including collecting and analysing data and meeting the needs of known users
* create, re-use, revise and re-purpose digital artefacts for a given audience, with attention to trustworthiness, design and usability

[Read the full KS3 computing curriculum](https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/239067/SECONDARY_national_curriculum_-_Computing.pdf)

## Scotland Curriculum for Excellence

#### Technologies

* I can describe different fundamental information processes and how they communicate and can identify their use in solving different problems. (TCH 3-13a)
* I am developing my understanding of information and can use an information model to describe particular aspects of a real world system. (TCH 3-13b)
* I can describe in detail the processes used in real world solutions, compare these processes against alternative solutions and justify which is the most appropriate. (TCH 4-13a)
* I can informally compare algorithms for correctness and efficiency. (TCH 4-13b)
* I can describe the structure and operation of computing systems which have multiple software and hardware levels that interact with each other. (TCH 3-14b)
* I can explain the overall operation and architecture of a digitally created solution (TCH 4-14b)

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

## Northern Ireland Curriculum

#### Science and technology - technology and design - statutory requirements, KS3

* demonstrate practical skills in the safe use of a range of tools, machines and equipment;
* research and manage information effectively to investigate design issues, using mathematics and ICT where appropriate;
* show deeper understanding by thinking critically and flexibly, solving problems and making informed decisions, using Mathematics and ICT where appropriate;
* communicate effectively in oral, visual (including graphic), written, mathematical and ICT formats showing clear awareness of audience and purpose.

[Read the full technology and design statutory requirements](https://ccea.org.uk/downloads/docs/ccea-asset/General/Statutory%20Requirements%20for%20Technology%20and%20Design%20at%20Key%20Stage%203.pdf)

#### Digital skills curriculum KS3

**Become a digital maker at KS3**

* Design a digital solution for a problem using an appropriate method;
* Build a solution based on their design using appropriate tools and techniques;
* Review or test the solution against their original plan;
* Evaluate their solution;

**Becoming a digital worker at KS3**

* Use applications to create products with thought given to both the audience and the purpose through the use of digital design;
* Troubleshoot basic problems with their digital technology.

[Read the full digital skills curriculum](https://ccea.org.uk/learning-resources/digital-skills-hub/key-stage-3-digital-skills-curriculum)

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

**Level 5**

Pupils should:

* create more sophisticated coding projects using a broad range of commands and more than one platform; and
* solve a more complex problem using commands in a programming environment.

**Programmable devices (such as Parrot Drone, MicroBit or Sphero)**

* as a class look at and talk about examples of coding projects, including using multiple ‘if...then’ and ‘if...then...else’ commands, variables, sensors, events, operators and comparators;
* recognise how they can decompose these projects;
* in small groups, plan their own coding project, demonstrating a clear sense of purpose and audience, showing understanding of abstraction by deciding what details they need to include and what they can leave out, working out what different parts of the program must do and using logical reasoning to discuss and compare the commands that are required for their algorithm and predicting the outcome;
* use a range of commands to create a project, including variables, operators and control statements such as ‘if... then...’ alongside the use of ’if...then...else’ and comparators;
* 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 the work of others evaluating process and outcome; and
* organise files and publish work online (if available) so that others can 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 4 - computation is the foundation for our digital world:

* I can decompose given problems and select appropriate constructs to express solutions in a variety of environments.
* I can plan and implement test strategies to identify errors in programs.

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

#### Digital competence framework

Progression step 4 - data and computational thinking - problem-solving and modelling:

* I can create a simple model or self-contained algorithm.
* I can identify the different parts of an algorithm to determine their purpose.
* I can detect and correct errors in algorithms.

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

## Code.org

#### CS Discoveries

Unit 1

Concepts included:

* problem solving
* inputs and outputs
* storing and processing information

[Read the full Code.org CS Discoveries curriculum](https://studio.code.org/courses/csd-2021)

## USA CSTA Standards

#### Grades 6-8

* 2-CS-03 - Systematically identify and fix problems with computing devices and their components.
* 2-AP-10 - Use flowcharts and/or pseudocode to address complex problems as algorithms.
* 2-AP-17 - Systematically test and refine programs using a range of test cases.

[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.