

Laying Firm Foundations

A conceptual approach to programming

Training Notes



Background

Most pupils, arriving in secondary schools will be familiar with the notion of algorithms as a sequence of steps. They will also have had some exposure to writing simple programs. They will probably have encountered sequencing instructions, and used repetition and conditional expressions. At Key Stage 3 we need to emphasise that algorithms work hand in hand with data structures. Pupils will have less exposure to this idea. To provide both continuity and progression at KS3, practical programming needs to provide repeated exposure to the same constructs and basic structures (variables and arrays) in a variety of contexts. The practical outcomes should reinforce the key constructs they are familiar with from primary schools but as programs become more complex, so there is a need to emphasise decomposition as a technique by which complex problems can be broken down into their constituent parts. Structured programming can then introduce procedural abstraction whereby code sequences can be defined as named functions and procedures. Once defined, the detail can be hidden; a function or procedure will perform its task when called by name. Structured programming – developing the idea of top-down design, decomposition and abstraction – is a key element in equipping children to tackle more complex challenges.

Once familiar with these foundations, the extra demands presented by text based programming can be addressed. There is a move to introduce text based programming very early (even in primary schools) but visual languages provide a much lower floor of entry, and can be used to good effect right through secondary school. Presenting the same constructs in a visual and text based language can help make conceptual connections.

Learning to program is difficult. There is a heavy cognitive load. This Unit is the first of two with a strong emphasis on concepts that run, like a golden thread through approaches to all programming challenges. It urges a staged approach, with each activity having limited goals to minimise potential confusion. It uses ‘toy environments’ to focus on specific constructs or concepts, whilst providing motivating contexts.

The aim of the day

First we want to consider the bigger picture – why teach programming at all? Secondly we need to consider the fact that students will come from primary schools with very different experiences and exposure to Computing. Thirdly we want to consider how to address some of the issues raised by text based programming.

The starting point for most teachers new to Computing will, inevitably be the National Curriculum ... and this will take time to embed. It's therefore important that teachers at KS3 are aware of expectations in the primary curriculum, which are only likely to be partially met in the near future. Implementing algorithms as simple programs is an expectation right from KS1. At KS2 some of the key concepts underpinning programming are made explicit – decomposition; sequence, selection and repetition; variables; debugging by thinking logically. These lie at the heart of the mental toolkit we wish to develop in children throughout their time in school. Looking at KS3 there is the first mention of text based programming languages, data structures and modular programs. Whilst much of the NC is very slim on detail, when it comes to programming it is explicit and provides a clear developmental structure. The bigger question is why? – Why teach programming at all?

In 2012, a very influential study, by the Royal Society made the case for a new Computing curriculum. Eminent biologist, and Nobel prize winner, Sir Paul Nurse in the introduction, “It is becoming increasingly clear that studying Computer Science provides a ‘way of thinking’ in the same way that mathematics does there are strong educational arguments for looking at how we introduce the subject.” Developing that ‘way of thinking’, which we commonly refer to as Computational Thinking is our rationale.

A simple way to get this across is to suggest to children that thinking like a computer scientist is a special way of looking at the world. Very soon, they will be seeing algorithms everywhere. They will be breaking problems down, spotting patterns and recognising this weird thing called abstraction, again and again and again. Computational Thinking rests on several conceptual pillars, so teaching programming involves a conceptual approach. These concepts can be easily transferred from one programming language to another, but also to approaching problems on other, non-programming contexts.

A Short History Lesson

For many ICT teachers, teaching programming will be a new challenge. Even those experienced at teaching Computing to older students will face new challenges making the ideas accessible to younger pupils. Over the next few years new subject pedagogy will emerge from this collective experience. A key aim of these sessions is to encourage discussion around what we are doing and what works best. But there is no need to start by re-inventing the wheel. There is a rich pedagogical history on which we can draw.

Logo was the first educational programming language, developed in the 1960's and 70's by Seymour Papert. Logo was used to write programs for floor turtles, the forerunner of many of the devices we see in Primary schools today, such as BeeBots. Papert worked at the Massachusetts Institute of Technology (MIT) and wrote a seminal book, *Mindstorms: Children, Computing and Powerful Ideas*. Many of these ideas shape our approach to teaching today.

Mindstorms appeared in 1980 when educational software was just beginning... .. and yet Papert could already see some of the dangers inherent in 'drill and skill' approaches. It's quite remarkable the insights Papert had. In the early days people were dismissive when he suggested computers could be instruments for children's learning, yet now we find ourselves standing on the shoulders of this educational giant; "In many schools today, the phrase 'computer-aided instruction' means the computer is being used to program the child...", wrote Papert in 1980, "...in my vision, the child programs the computer and, in doing so comes into contact with some of the deepest ideas from science, maths, and the art of intellectual model building." He goes on to say that "building knowledge structures' ... happens where the learner is consciously engaged in constructing an entity ..." In essence he is arguing that children learn best by doing – something that should inform our approach today. Children cannot be taught didactically to program, but need to be presented with challenges through which they develop their own understanding.

Mitch Resnick was a doctoral student of Seymour Papert. Like Papert he was involved in the early development of Lego Mindstorms. Both appreciated how Lego helped develop creative play and how constructing things helped shape children's 'knowledge structures'. Resnick is Director of the Lifelong Kindergarten Group at MIT. They developed Scratch, which launched in 2003. Scratch has been a game changer in education. There will be few Primary children who haven't had some form of contact with it. It has a very low floor of entry. It is a visual language, which removes the need to type commands. You can see the analogy with Lego in the way sequences can be snapped together.

Scratch has spawned many variants – which have often extended its capabilities. One such variant, first used to teach undergraduates at the University of Berkeley in California was BYOB (Build Your Own Blocks). Using this provides a seamless transition for pupils who already have some experience of Scratch at Primary school, but it is far more powerful. There is an online version, "Snap!", but this currently doesn't have all the functionality required for our projects, primarily the need to set multiple instruments. This situation will change with time.

Using a visual, block based language as a starting point allows for both continuity and progression at KS3. Whilst many pupils will have had exposure to activities in Scratch it is likely that they will all benefit from an emphasis on concepts that help to develop the mental toolkit required for problem solving. Identifying those tools is the over-arching aim of this Unit.

This Unit is the first of two that introduce a conceptual approach to learning how to program. Keep in mind the main purpose of the day – to engage experienced teachers with some of the deeper ideas in Computer Science they may not be familiar with. Most will be adept at programming, so although the exercises are aimed at beginners, the emphasis is on how they can be used to focus on specific aspects in a staged approach, the pedagogy involved and, crucially the recurrent conceptual themes which we wish to bring to the fore. Whilst many may be experienced programmers, they may also be self-taught and not familiar with a conceptual approach. The aim is to empower the experienced teachers, providing them with material they can deliver and use with beginner programmers in shorter training sessions.

There are key exercises which should be completed and supplementary material to discuss. The pace should be fast, with the assumption that the audience are experienced teachers, probably already teaching to GCSE level, with some familiarity with the concepts of Computer Science and programming. As such they may benefit from working through some of the more challenging extension activities. Timing is therefore very much in the hands of the presenter. Sometimes it will be sufficient to part complete an activity so teachers 'get it' and can discuss how it might be used.

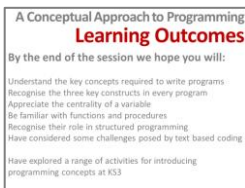


The Unit was developed with a full day in mind, but it should be possible to deliver it in a shorter period. Judgement is required and the timings below are indicative, to help with planning. Always be flexible and encourage discussion and engagement. Details of each activity are given in the individual Session Notes. Further guidance on the narrative, slide transitions and animation can be found in the slide notes.


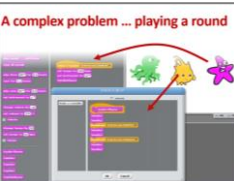


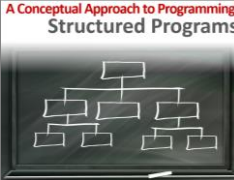
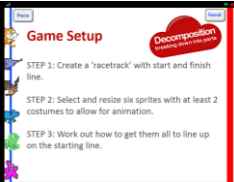
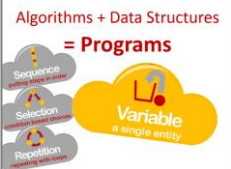

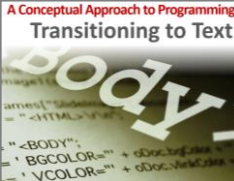

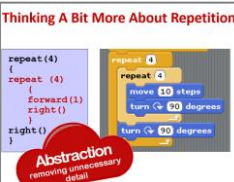



Indicative Timetable

Each of the three sessions covered aims to include enough material for attendees to take away and run as separate half day sessions for beginner programmers.

Above all else, remember that the aim is to empower the attendees to offer similar sessions to colleagues. The central goal is to build confidence and a desire to spread the message. It should be inclusive, enjoyable and embody the CAS ethos of collegiality: There is no 'them', only us!

The trainer's presentation is broken down into 5 sections, with formal inputs and practical activities outlined below:

 <p>By the end of the session we hope you will:</p> <ul style="list-style-type: none">Understand the key concepts required to write programsRecognise the three key constructs in every programAppreciate the centrality of a variableBe familiar with functions and proceduresRecognise their role in structured programmingHave considered some challenges posed by text based coding <p>Have explored a range of activities for introducing programming concepts at KS3</p>  <p>20 minutes</p>	<p>Establishes key outcomes from the day for teachers (5 mins).</p> <p>Quickly considers the requirements of the national curriculum from KS1 up (5 mins).</p> <p>Sets the session in the context of the key educational goal: developing computational thinking (10 mins).</p> 
---	---

  <p>75 minutes</p>	<p>Considers the history and impact of visual languages putting pedagogy to the fore in subsequent activities (10 mins).</p> <p>Practical introduction to BYOB (10 mins).</p> <p>Making music: A practical activity to introduce procedures and consider the concepts involved (15 mins).</p> <p>Playing a round. Practical challenge to emphasise pedagogy of learning through doing (20 mins).</p> <p>Reinforcement activities using LightBot (10 mins).</p>	 
  <p>90 minutes</p>	<p>Formal input introducing the key role of a variable, considering the most effective way for children to embrace the concept (20 mins).</p> <p>Drawing a squirrel: Short practical activity to establish the concept (10 mins).</p> <p>Discussion on decomposing a larger problem (10 mins).</p> <p>Wacky Races: Practical activity bringing concepts together (20 mins).</p> <p>Hungry Frog: Analysing scripts and suggestions for scaffolding projects (15 mins).</p> <p>Networked Pong: Considering extension challenges (10 mins).</p>	 
   <p>90 minutes</p>	<p>Introductory discussion about the syntax barrier, pedagogy and motivational contexts (10 mins).</p> <p>Introduction to RoboMind (5mins).</p> <p>Pass The Beacons: Short practical challenge emphasising code layout (10 mins).</p> <p>Teaching Robo To Walk: Formal input emphasising cognitive difficulties involved in combining constructs, decomposition, abstraction, code layout and flow charting as a visual tool (20 mins).</p> <p>Under The Hood: Explanation of maps (10 mins).</p> <p>Find The Spot: Input distinguishing between counter-controlled and condition -controlled loops followed by practical challenge. Can be extended to develop a maze runner solution (20 mins).</p> <p>Developing procedures. Input illustrating links between visual and text based languages, developing a procedure, highlighting parameters (10 mins).</p>	 
 <p>10 minutes</p>	<p>A short discussion to promote classroom research and encourage reflective practice.</p> <p>Draw out suggestions for potential research areas and mention possible techniques.</p> <p>Ends with a quick promotion of the BCS Certificate in Computer Science Teaching</p>	

When someone books to attend the training session, send a prompt acknowledgment informing them when final confirmation and further details will be sent. Set a cut-off date, at which point you decide if there are enough bookings to make a viable session.

Once you have enough people booked, contact them again with brief details and suggested prior reading. Although not essential, by suggesting some prior reading you are indicating that this is in depth CPD which requires some commitment on the part of the attendees. It also gives you a chance to establish some dialogue with attendees prior to the event. With a week to go, you could mail a reminder and enquire about the reading and whether it would be useful for teaching. This helps keep the attendees focused on the event.

Prior Reading

This Unit has a practical flavour. To allow it to focus on pedagogy and avoid too much time getting used to software it would be advantageous for all attendees to have had some initial experience of BYOB and RoboMind before the event. If they are bringing their own computers, both should be installed.

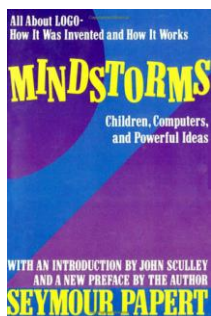
It's worth noting that BYOB now runs 'in the cloud'. This version is known as Snap! It has certain advantages; primarily no installation is required and work is accessible from any computer online. But it also has disadvantages, particularly when used with Internet Explorer (where sound may not work) and will require a user ID to upload any media files. At the time of writing, Snap! did not support multiple instruments, which are essential to our main project. We therefore use a local installation of BYOB in the examples. This situation will change with time so check the development of the online version. In the sound command drawer look for the instrument block. If it is present and attendees wish to use Snap! suggest using Chrome as the browser and check that sound works prior to the event. Snap! is at <http://snap.berkeley.edu/>, whilst BYOB can be downloaded from goo.gl/jsrL2g. A free GPL version of RoboMind can be downloaded from goo.gl/gCgzNG.



Paul Curzon, from QMUL, editor of the free magazine and CS4FN website, has written an excellent introductory book, "Computing Without Computers". It introduces many key ideas in Computing in a child friendly way, often by way of analogy. Although the whole book is recommended, chapters 3 to 7 would make ideal introductory reading for this session, introducing a host of ways to explain the key constructs of sequence, selection and repetition, along with ideas to introduce variables and accessible explanations of functions and procedures. They can be downloaded from goo.gl/obpAUq.

Further Reading

You can watch Mitch Resnick arguing why we should teach kids to code in his TED talk from 2012. It is available at goo.gl/OVJGCa. The session also uses a clip from Alan Kay's earlier TED talk "A Powerful Idea About Ideas". This too is worth watching in full, to get an understanding of the pedagogy and potential inherent in his approach to learning. It is available at goo.gl/Qe8nMb.



To gain a deeper insight into the pedagogy that lies behind Seymour Papert's idea of Constructionism reading Mindstorms: Children, Computers, and Powerful Ideas is a must. Although the book is dated, dealing with the programming language Logo, how it was invented and how it works, the insights Papert offers into how children construct knowledge are as valuable today as when they were written. It is a seminal book that every teacher should try to read at some point. If the book is a little intimidating, there is a collection of his essays on his website, which make a good starting point to find out more about this remarkable man: <http://www.papert.org/>.

The Training presentation is designed to support a one day session, delivered to CAS Master Teachers and other curriculum champions. However, the timing and pace will, to a large extent be determined by the experience of the attendees and their familiarity with the software. It is envisaged that it will be delivered to experienced teachers. There is an expectation that the programs at the heart of the key activities will be created, but there is flexibility here. It could be delivered in a shorter session if examples were 'talked through' rather than built. Equally, if the attendees are very experienced, some of the extension challenges could be programmed.

It is envisaged that attendees will take the material and deliver shorter sessions, either as half day, twilight or CAS Hub inputs. It is expected these will take longer to cover each activity as the material will be unfamiliar to teachers new to Computer Science. Please find time to discuss with attendees possible ways to use the material and encourage them to offer further sessions in their locality.

Resources

All supporting material is available for download, corresponding to each session in the Unit.

The Unit includes a full Training Presentation and these accompanying Training Notes.

Further material corresponds to each Session in the Unit and includes;

- a separate Session presentation to support all the activities covered in the session
- Session Notes explaining the resources for each session
- separate activity handouts, suitable for classroom use

If you intend to use the material in shorter sessions, please use the Session presentations as these are tailored to self-standing delivery.

Half Day / Twilight CPD Sessions

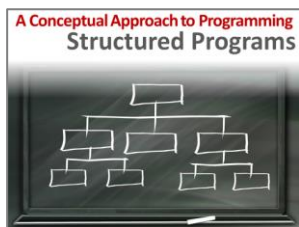
It is suggested the material could be delivered as 3 separate shorter sessions:



Concepts and Continuity: Introduces BYOB/Snap!, and has a focus on the key concepts involved in programming and the core idea of named procedures.

Structured Programs: Emphasises the importance of a variable, the use of top-down design and suggests ways of scaffolding projects.

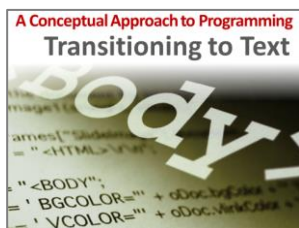
Transitioning To Text: Provides a series of tightly structured exercises that avoid variables, focusing on key constructs, code layout and understanding syntax. Continues with exercises developing named procedures.



Of course, Master Teachers and other trainers can combine sessions and activities as they feel best fit the local circumstances. Do note however, the prerequisite knowledge for each of the activities.

The Unit has been written with progression in mind and, as such might best be delivered as a sequence of three afternoon sessions for beginner programmers.

Many single programming activities are short enough to introduce at CAS Hubs or worked through in a school departmental meeting. Whatever ways you choose to disseminate the ideas please remember the aim is to develop teachers appreciation of CS concepts, not just demonstrate an activity. To that end, we hope you will find them useful.



A new subject offers lots of potential for research on the part of teachers. Throughout the activities in this unit there is encouragement to consider issues, reflect on classroom practice and engage in action research. Not only does it contribute to practitioner's professional development, but can provide a key part of the evidence required for teacher accreditation via the BCS Certificate in Computer Science Teaching.

Preparation required:

Publicity for the BCS Certificate available for all attendees.

Familiarity with the questions to be posed and consideration of how to facilitate discussions.

Points To Ponder



Scattered throughout the Training Presentation are 'Points To Ponder' slides. Usually at the end of an activity, they have a dual purpose, both practical and professional. From a practical point of view, by posing a question for short discussion, they provide the presenter with a few minutes to prepare for the next activity and gather their thoughts. Moreover, they encourage attendees to converse with each other. The aim is not to engage in a lengthy discussion, but to plant ideas that could be pursued as classroom research. Near the end of the presentation, we hope you will raise the value of such action research.

In this Unit, the questions posed at the end of (or during) an activity are:

Concepts and Continuity: Visual languages have proved a 'game changer'. Why do children seem to grasp a visual language better? They have a low floor of entry allowing young children to express their computational thinking. But many children still founder on the transition to text. Why? Is a different pedagogy required to introduce them?

Structured Programs: Paired programming is a popular approach for developing pupil capability. Do teachers have other ideas? If so, how can we evaluate them? This is a fertile area for action research – a theme we return to at the end of the Unit. The questions posed are; Scaffolding projects, reading code and paired programming are all techniques to aid learning. How do we know what works? How can we contribute to subject pedagogy?

Transitioning To Text: After the introductory activity, pause and try to draw out recognition of the challenge the syntax barrier poses. Restricted environments, such as simple html markup, and RoboMind which has no variables, help focus on specifics and lower cognitive load. Questions posed are; What lies behind the syntax barrier? What facilities do 'toy' environments like RoboMind offer to help with the transition to text? Can you suggest helpful tools in other languages?

Transitioning To Text: After the main walking activity, pause again. Notice that, up until now, when introducing text based languages we have focused entirely on the algorithmic constructs. None of our Robo scripts has involved variables. In fact the free version of RoboMind doesn't allow variables to be defined (except as parameters in a procedure). The cognitive load involved in getting to grips with programming is huge. Constructs, data structures and now syntax. The key to handling complexity is to decompose the skillset into smaller parts, finding contexts that allow you to focus on a subset of the overall cognitive load. In the sequel to this Unit, How Computers Do Stuff we look at another introductory environment that allows you to scaffold the transition to text in other ways. There is no one correct answer to managing this transition, so pose these open questions; Learning to program is challenging. Can we decompose that challenge? What different elements contribute to the cognitive load?

Class Based Research

Why? Before concluding the session, it is worth pausing to prompt a discussion about the value of classroom research both to support continuing professional development and to inform the wider teaching community. Use the terms “teacher enquiry”, “action research”, “reflective practitioner” and “practitioner research”. Teaching computer programming is a relatively new activity. Many teachers are experiencing it for the first time. Those with experience can offer valuable support, advice and information to other colleagues. There are lots of opportunities for colleagues, old and new to contribute to an emerging pedagogy, particularly in this field.

Inform attendees of the CAS Teacher Inquiry in Computing Education project. It provides a forum and focus for research in the field of computing. It can be accessed from the home page of the CAS website, under the link to projects: <http://www.computingschool.org.uk/>

Research an area in which teachers have ready access to the data but be aware of researcher bias and local ethical considerations. Don't try to prove something you believe – always think that you are “bringing a better understanding” of the situation – it helps avoid bias.

What? Emphasise the key words such as “find out”, “evaluate” and “compare” to spark ideas. Each of the following give examples for a possible focus for research:

Find out about pupils preconceptions of why they are learning computer programming

Evaluate the use of a regular starter based on tracing (dry running) or reading code.

Compare school data about the pupils with their performance in programming sessions.

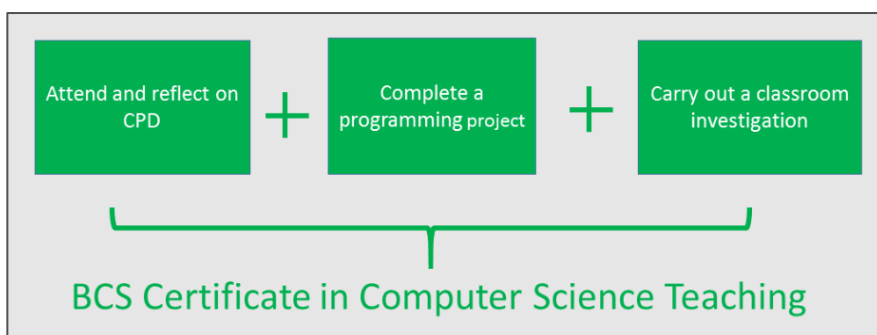
Design, implement and evaluate a module to support primary transition.

How? Briefly tell the audience of different methods. Choose one of the examples from the suggestions above, or those raised in ‘Points To Ponder’ and discuss methods which might be used.

Remember – all research has ethical considerations. Remind them that if they embark on a University course then they will be governed by their ethics policy. Similarly, if completing the BCS Certificate then there is an explicit ethical requirement.

BCS Certificate in Computer Science Teaching

End with a reminder of the BCS Certificate in Computer Science Teaching – ensure hand-outs are available. One part of the evidence for the award is classroom research undertaken by the teacher. The other two areas involve attending CPD – like today, and completing a manageable programming project.



If that sounds a little daunting, remind teachers they can opt for either an independent or guided route. The latter means they have support from a mentor who can help guide them through the requirements. It is a valued award, giving professional recognition, accredited by the BCS, The Chartered Institute for IT. More importantly, it's designed to help teachers in the work they are developing in school. Encourage teachers to seek support from their schools to gain accreditation. Ensure they raise it as part of the performance management cycle. More information is available at: <http://www.computingschool.org.uk/certificate>.