Programming for four-year-olds

No assembler required

How to teach computer science in nursery school

Aug 1st 2015 | Boston | From the print edition

COMPUTING has always been a youngster’s game. The founders of Apple, Facebook, Google and Microsoft were in their teens or 20s when they started the businesses that made their fortunes. But even by the standards of Messrs Jobs, Zuckerberg, Brin, Page, Gates et al, the code jockeys of a programming language called KIBO are wet behind the ears.

KIBO is designed for those aged four to seven. Instead of arranging, as an older programmer might, a set of constants, variables, operations and expressions, all written in something resembling English, into a logical sequence, a KIBO programmer arranges wooden blocks that carry stickers bearing symbols. These symbols tell a plastic robot what to do next. A straight arrow means “move ahead by one foot”. A curved one means “turn in the direction in which the arrow is pointing”. Two semicircular arrows pointing towards each other’s tails means “perform the previous instruction again”—a command that is particularly important, because it introduces neophytes to the concept of recursion.

For more complicated commands, such as “shine a light”, “make a noise” or “wait here until I clap before moving again”, KIBO offers a library of subroutines that a fledgling programmer can call on, just as a grown-up might. And because programs written in languages used by grown-ups usually have specific “begin” and “end” instructions, a block bearing a big green dot means “start” and one with a big red dot means “finish”.

To compile and execute a newly written KIBO program, all the junior software engineer must do is lift the robot up and scan it along the row of blocks (see picture above), so that it can read bar codes on the labels. These tell it what symbols those blocks carry. Then, placed on the floor, the robot is off—to attack the family pet or cause whatever other mayhem has entered the mind of
the programmer.

Wheels on fire

KIBO is thus an entertaining toy. But it is also a toy with a purpose. This is to increase the supply of people who are genuinely computer-literate—for, despite what they may think about themselves, most so-called digital natives of the internet generation are not. They are, it is true, whizzes at operating the devices technologists have thrown at them, but few have much idea what is going on under the bonnet.

Many parents and teachers want to change this, by encouraging children to learn how to program at the same time as they are learning how to read and write, and add and subtract. Some governments, such as those of Britain, Estonia and Finland, are making programming part of the school curriculum. KIBO, brainchild of Marina Umaschi Bers, a psychologist at Tufts University, near Boston, is the logical conclusion of this train of thought. Its aim is to engender computer literacy in children who are still at nursery school.

Dr Umaschi Bers is not alone in that quest. KIBO, made by KinderLab Robotics (of which she is chief science officer when she is not doing her day job), is unusual only in that its instruction set is so tied to physical objects. Other toys being developed to teach young children the rudiments of programming use not wooden blocks but blocks of code, presented as icons of various sorts on the screens of tablets, smartphones and even old-fashioned PCs. Instead of being scanned, these instructions are uploaded wirelessly to the robots they are intended to control—robots that come in all sorts of shapes and sizes.

Some, like Vortex (a wheeled device that resembles a flattened motorcycle helmet) and Dash (a tetrahedron of spheres which, besides moving around at its programmer’s command, can also play tunes on a glockenspiel), are, like KIBO, designed mainly to scuttle across the living-room floor. Others, though, are heading in a different direction.

One such is Hackaball, which is being developed by two firms, Made By Many, and Map, that are based in London. Hackaball is an actual ball, made of rubber and silicone, and 10cm (4 inches) in diameter. Its contents, including solid-state lights of various colours, a motion detector, a microphone and a loudspeaker, are held in a springy silicone housing that acts as a shock absorber. This means the ball can be thrown against walls and floors without damaging the hardware inside it.

Hackaball acts as a *tabula rasa* on which a programmer can write whatever comes into his imagination. In tests, one group of children used it to play Truth or Dare, based on the colour it turned when it was thrown and caught. Another lot invented Bomb Detector, a game like Pass
the Parcel in which the smallest shake would cause the ball to vibrate, meaning that player who had shaken it was out. A more practical child, meanwhile, programmed a Hackaball to be an alarm clock, vibrating and making noise at a preset time.

Starting from Scratch

Toys like Vortex, Dash and Hackaball use a variety of programming languages to encode the instructions that control them. These include Scratch, Blockly, Hopscotch and WeDo. Some of these languages are proprietary (WeDo, for instance, belongs to Lego). Others are not. All have in common, though, a rejection of the vexing, obscure delimiters of syntax, such as brackets and semicolons, that litter the code of more conventional languages.

The oldest of the group, and the one that inspired the others, is Scratch. This was developed in 2005, by Mitchel Resnick of the Massachusetts Institute of Technology (MIT). It was given a makeover in 2013—and then, in 2014, released in a simplified version called Scratch Jr.

The result is a crucial bridge by which an infant programmer can cross from the physical world of controlling toys to the virtual one of controlling cyberspace. Scratch can be, and is, used to run robots. But it also encourages its users to create animations and games that manipulate pixels on a screen rather than objects on the floor.

A Scratch programmer can thus design and animate an icon called a sprite (a smiling cat that walks like a human being is a popular choice for beginners), and send it on adventures. The programmer chooses, from a set of menus, coloured blocks that represent various subroutines. He then drags them, with finger or mouse, across the screen to assemble them in the order he wishes. “Motion” subroutines tell the sprite how and where to move. “Looks” subroutines regulate its shape and colour. “Sound” subroutines give it voice. “Control” subroutines govern things such as recursion, causing it to repeat an action. “Events” subroutines include the start and stop commands. And so on.

Scratch Jr, which has been given a restricted set of subroutines and uses only positive integers for counting (because young children have difficulty with the concept of negative numbers) has proved particularly popular. In the 12 months since its release, 1m copies of it have been downloaded. But Hopscotch (which also has a restricted set of subroutines) and Blockly (which, unlike Scratch, is fully open-source, and can thus have its underlying code tweaked by more advanced programmers) are also doing well.

For proponents of early computer literacy this all looks encouraging. It does, though, remain to be seen whether it will turn out to be a fad—for this is not the first time the gospel of child-friendly computing has been preached.
In 1967 Seymour Papert, a researcher at MIT who went on to teach Dr Umaschi Bers and Dr Resnick, created a programming language called Logo that could be used to control a turtle-shaped robot. Logo prospered for a while but has now more or less faded from view. Probably, it was an idea ahead of its time as, in the 1960s and 70s, computers were still exotic creations with which mere mortals rarely interacted during their daily lives. These days, however, the risk is precisely the reverse. Computers have become quotidian, and also so reliable that, for most people, the idea of programming one might seem as bizarre as that of servicing one's own refrigerator or washing machine.

Whether KIBO and the other programming languages will do the job their inventors intended thus remains to be seen. Cynics might ask, when all is said and done, who actually needs to know how a computer works? But if the world's first software trillionaire, sometime in the early 2030s, fondly recalls her childhood japes waking up the family dog by teasing it with a wheeled robot which she had programmed for the purpose, remember that you read it here first.

From the print edition: Science and technology