code-it.co.uk

Solving Algebra using Inverse

Planning Aim Turn a sequence based algorithm into code that asks and solves simple algebra questions using inverse rule

Computational Thinking Concepts

-All programming is an **algorithm** turned into code. Not all **algorithms** become code though.

-**Algorithms** are the thinking before you code. Where the user thinks what they want to happen and what **order** they want it to happen in.

-Programming is where we turn that algorithmic thinking into something that works on a digital device

-**Thinking with variables** in place of numbers which has a real connection to the use of variables in algebraic maths

-**Generalise** the solution for subtraction into one that works with other operations

Programming Concepts

-**Sequence** The critical order of instructions which is very important in programming

-**Selection** Meeting a condition or not triggers different outcomes

-**Variables** A named space that holds something (number in this case) that can be varied or changed

Maths Concepts

Some algebra problems can be solved by using the inverse operation on the other side of the expression

http://www.mathsisfun.com/algebra/introduction.html

Learning Path →

1, Understand addition algebra algorithm	2, Convert addition algorithm into code	3, Generalise solution to work with other operations	4, Find a graphical solution that moves numbers	6, Evaluate Solutions

Pupils 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**

Debugging -Matching the algorithm to the program is key here. Point pupils back to the algorithm as this holds the key to build and debug the program.

Prior Knowledge

-It is advisable to use selection in a quiz context first before attempting this module.

-It also helps if pupils have worked with numbers in a programming language prior to attempting this module such as *how to teach computer to do maths*.

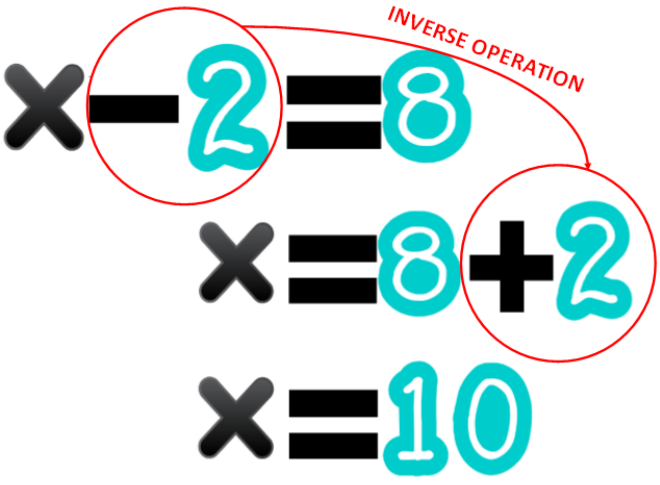
You can find a well resourced quiz module and numerous maths modules including the one mentioned above in *How to Teach Primary Programming Using Scratch*

Maths Use

Whilst as a computing science advocate I would like your pupils to have the joy of thinking through and creating this program there is a complete version for maths purposes at https://scratch.mit.edu/projects/105772342/

Solving Algebra using Inverse page 2

1, Understand addition algebra algorithm



It is advisable to link this module to maths learning that expresses equations using the inverse method. Pupils who have calculated these in maths and written a program to do so in computing are far more likely to remember and understand the concept.

The pupil algebra sheet has four simple equations to express using the inverse method. If you are working on this in maths pupils may not need these. If pupils are not working on this at the moment it is worth them calculating these first and you going through one or two with the class as a reminder.

Go through the questions on the sheet

Which part of the expression is already a variable?

X is the right answer here as it represents an unknown quantity.

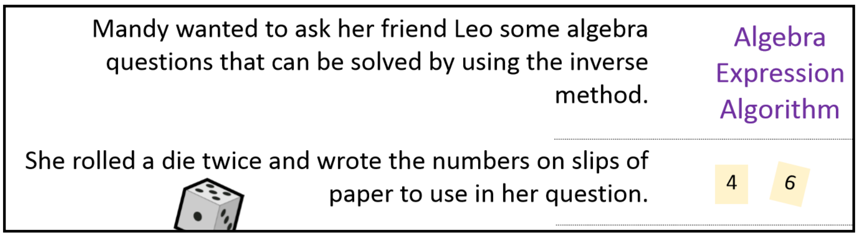
Which other parts of the expression change in every calculation?

The two numbers change in every expression. These are easily replaced by variables.

You will need post it notes, a large felt tip pen & a die.

Understanding the addition algebra algorithm

Project the algebra expression algorithm and move to slide 2.



Mandy wanted to ask her friend Leo some algebra questions that can be solved by using the inverse method.

Algebra Expression Algorithm

She rolled a die twice and wrote the numbers on slips of paper to use in her question.

Explain that Mandy wanted to ask her friend Leo some algebra questions that can be solved using the inverse method. Roll a die twice and write outcomes on two separate post-it notes. Ask the class why she might use a die to generate the numbers. You are looking for the response that it adds an element of random to the question so most will be different?

Create the same question on the slides on the board using the post-it notes and after calculating it with the class write down the value of x on a new post-it note. Make a big show of writing x on one side of the post-it note. Ask the pupils what would be a good variable name for the numbers, accept all sensible solutions but name them a and b on the other side of the post-it notes.

Solving Algebra using Inverse page 3

1, Understanding the addition algebra algorithm continued

Ask a pupil to write their answer on a post-it note. Make a big show of comparing these. Ask the class if they are the same? Are they equal?

Leo wrote his answer down on another slip of paper. 10

Mandy compared Leos answer with her answer, she had worked out previously. She had decided that if they were the same she would say correct and if they were different she would say wrong. 10 = 10

Ask pupils where we have compared things to see if they are the same before? You are looking for any selection work although the maths quiz is the most obvious example on code-it.co.uk.

2, Convert addition algorithm into code

Move pupils onto computers using either Scratch 1.4 or Scratch 2.0. Display or print out and give slide 2 of the algebra expression algorithm so that all pupils have access to this. Remind pupils that their job is to convert this algorithm into Scratch code.

Many pupils start without referring to the algorithm and the best way to help them initially is to point them back to the algorithm. Ask them which part they are trying to solve? Get them to read their section aloud. Ask them to explain what their section is trying to do? Can they find the key words in the section? Underneath is a version with key phrases to help you help them identify what is important.

She rolled a die twice and wrote the numbers on slips of paper to use in her question.

In our program what do the slips of paper represent? Answer variables

She then thought of an algebra expression using these numbers and ***worked out*** the ***right answer*** using the inverse operation.

How can you get the computer to work out the right answer using the variables?

Mandy ***presented her question*** to Leo using her slips of paper and the flipped answer card

Can you ask the question using the variables?

Leo wrote his answer down on another slip of paper.

How can you collect in the answer?

Mandy ***compared*** Leos answer with her answer, she had worked out previously. She had decided that if they were the same she would say correct and if they were different she would say wrong

How can you compare things in Scratch to see if they are the same? Answer =

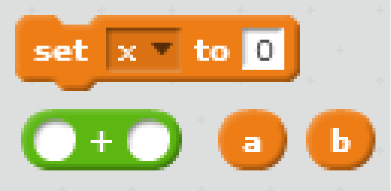
Solving Algebra using Inverse page 4

2, Convert addition algorithm into code (continued)

Depending on your class and their previous experiences some will struggle to complete this independently. For them we have two further levels of support.

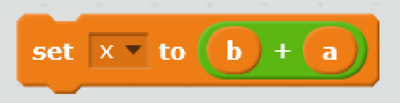
Algebra algorithm blocks hint sheet this has the blocks that pupils need to use, Most pupils who struggle won't need any more help than this.

Algebra algorithm code sheet this has the fully formed code to make the project (few pupils will need this) but it might be useful if pupils are new to programming



```
set x to 0
( ) + ( )
a
b
```

Algebra algorithm blocks hint sheet example



```
set x to (b + a)
```

Algebra algorithm code sheet example

3, Generalise solution to work with other operations

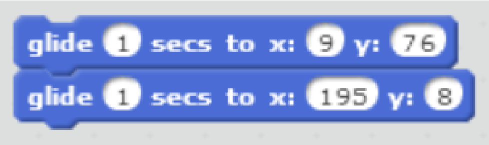
Pupils can copy their code and adapt it to solve other inverse operations. They will soon spot real issues when inverting multiplication and division as they encounter decimal fractions answers. A great extension is to challenge them to find a way to only deliver whole number answers.

4, Find a graphical solution that moves the numbers

Show pupils a working copy of the program without viewing the code. You can find one here. https://scratch.mit.edu/projects/105772342/
Some pupils will start working on a solution straight away some will need some hints to solve this. I recommend you look at the code before the lesson to understand one solution for making this work. You could also get pupils to decompose what needs to happen in detail for each sprite.

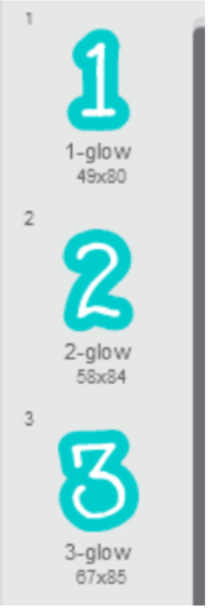
HINT

Note how the **glide to x and y block** can be used to rearrange the position of the numbers and or symbols



```
glide 1 secs to x: 9 y: 76
glide 1 secs to x: 195 y: 8
```

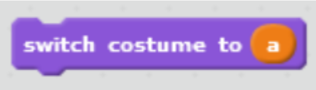
HINT You can organise code using either broadcasts or create your own blocks



1 1-glow 49x80

2 2-glow 58x84

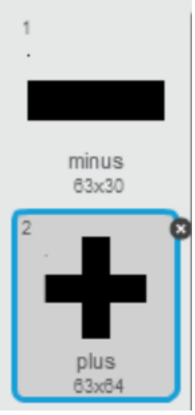
3 3-glow 67x85



```
switch costume to a
```

HINT

Note how multiple costumes can be switched between if their position is the same as their number. 1 in position 1 etc. You can then use a switch costume to block to change what the number displays to what was randomly chosen.



1 minus 63x30

2 plus 63x64

HINT

Use costumes to switch operations when using inverse

Solving Algebra using Inverse page 5

5, Evaluate Solutions

Why not finish the module by asking each pupil to report back on another pupils module.

Can they explain how it works?

What improvements would they make?

Can they spot any bugs?

Evaluating programs and algorithms is a key part of algorithmic thinking. How efficient is the code? Could the same effect be achieved in a better way using less code or in a way that makes it easier to adapt?

Disclaimer

This is a harder project. It is immensely satisfying when pupils solve or improve it but it is not for the first year of programming.

Talk for Computing
Assessment Questions

What were the key words in this part of the algorithm?

She then thought of an algebra expression using these numbers and worked out the right answer using the inverse operation.

Describe how you converted this part of the algorithm into code. Were there any problems you had to overcome?

Describe one bug that you have or have debugged? What strategy helped you fix it?

Generalisation is where you adapt an idea to work in a different way. Describe how you have used this computational thinking skill in this project?