Algorithms in pseudocode and flow diagrams

How do we solve problems? Think about working out the change that you would receive if you buy a chocolate bar for 25p and pay with a £1 coin. The solution is to break down the problem into steps. We know that we need to remember how many pence there are in £1. We also need to subtract 25 from 100. We then need to look at the change that we receive and make sure the coins add up to 75p. We can work out the stages needed to solve this problem and it doesn't matter if we're buying something that costs 5p or 95p. We're able to follow the steps to work out if we've been given the correct change.

So now let's think about the steps. We'll show things that we need to do, such as convert £1 into pence and then subtract the item price from 100 to calculate the change that we should receive.

Let's now make it a little more complicated and buy several different items. We might need more than £1. We need to work out how much the total is, how much we need to pay and then if we're due any change, how much it should be.

Flowcharts are a useful way of illustrating processes and steps, but if we make a mistake it can cause problems. It's important to correct problems before they happen while we are still planning the solution.

An algorithm is a step by step solution to a problem. Programming is all about solving problems. Sometimes these are simple and only need a few steps. Sometimes they're very complicated. Programmers turn the algorithm into code. There are many programming languages. Some are better suited to particular types of problem.

Flowcharts are helpful but it's also useful to write out the algorithm in code, which is rather like a program but not specific to one language.

Wouldn't it be great if we could speak one language and be understood by everyone?

Pseudocode is a generic code like language that can easily be written in any programming language. Let's look at the flowchart that we made earlier. Pseudocode is easy to read as it consists of simple statements but it also shows the programming constructs needed such as loops and selection.

Now let's look at some simple examples. Write the pseudocode for adding two numbers together, drawing a square or moving a sprite from one side of the screen to the other in steps. Now you try.

You also need to be able to work out what pseudocode will do and to spot any errors. Can you see anything wrong here? This prevents errors being transferred into program code which could lead to serious problems. Pseudocode allows programmers to see what the program should do which makes it much quicker to write the correct code.

We've looked at how flowcharts and pseudocode can be used to represent algorithms. Flowcharts are a useful way of showing solutions to whole problems, even when they're complex. Pseudocode is a useful way of communicating how a program solution might look in code. You need to know how to read pseudocode, check to see if it's correct, how to put errors right and how to write your own code from scratch.