**Designing an algorithm**

An algorithm is a plan, a logical step-by-step process for solving a problem. Algorithms are normally written as a flowchart or in pseudocode.

The key to any problem-solving task is to guide your thought process. The most useful thing to do is keep asking ‘What if we did it this way?’ Exploring **different** ways of solving a problem can help to find the **best** way to solve it.

When designing an algorithm, consider if there is more than one way of solving the problem. When designing an algorithm there are two main areas to look at:

* the **big picture** - What is the final goal?
* the **individual stages** – What hurdles need to be overcome on the way to the goal?

**Understanding the problem**

Before an algorithm can be designed, it is important to check that the problem is completely understood. There are a number of basic things to know in order to really understand the problem:

* What are the inputs into the problem?
* What will be the outputs of the problem?
* In what order do instructions need to be carried out?
* What decisions need to be made in the problem?
* Are any areas of the problem repeated?

Think about applying computational thinking: decomposition, pattern recognition, abstraction and finally algorithm. Once these basic things are understood, it is time to design the algorithm.

**Pseudocode**

Most programs are developed using programming languages. These languages have specific syntax that must be used so that the program will run properly. Pseudocode is not a programming language, it is a simple way of describing a set of instructions that does not have to use specific syntax.

**Common pseudocode notation**

There is no strict set of standard notations for pseudocode, but some of the most widely recognised are:

* **INPUT** – indicates a user will be inputting something
* **OUTPUT** – indicates that an output will appear on the screen
* **WHILE** – a loop (iteration that has a condition at the beginning)
* **FOR** – a counting loop (iteration)
* **REPEAT – UNTIL** – a loop (iteration) that has a condition at the end
* **IF – THEN – ELSE** – a decision (selection) in which a choice is made
* any instructions that occur inside a selection or iteration are usually indented

**Using pseudocode**

Pseudocode can be used to plan out programs. Planning a program that asks people what the best subject they take is, would look like this in pseudocode:

REPEAT

OUTPUT 'What is the best subject you take?'

INPUT user inputs the best subject they take

STORE the user's input in the answer variable

IF answer = 'Computer Science'

THEN OUTPUT 'Of course it is!'

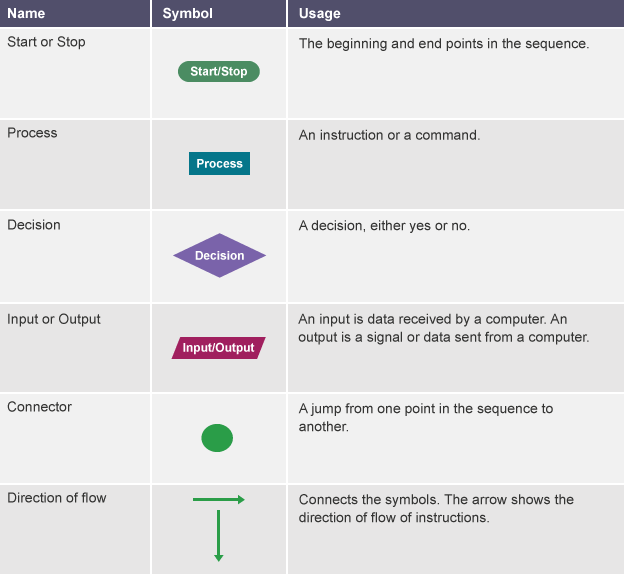
ELSE OUTPUT 'Try again!'

UNTIL answer = 'Computer Science'

**Flowcharts**

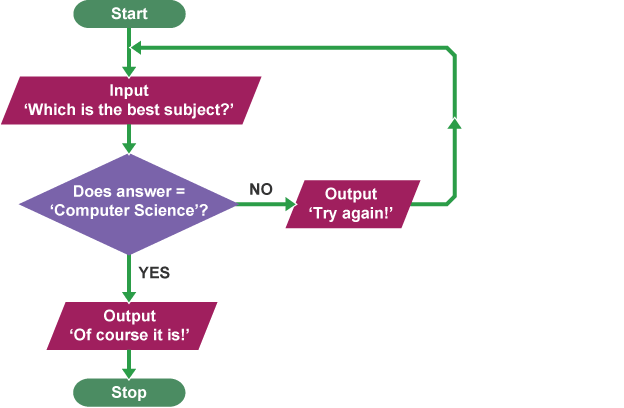
A flowchart is a diagram that represents a set of instructions. Flowcharts normally use standard symbols to represent the different types of instructions. These symbols are used to construct the flowchart and show the step-by-step solution to the problem.

**Common flowchart symbols**



**Using flowcharts**

Flowcharts can be used to plan out programs. Planning a program that asks people what the best subject they take is, would look like this as a flowchart:

**Boolean in programming**

**Boolean algebra** is used frequently in computer programming. A Boolean expression is any expression that has a Boolean value. For example, the comparisons 3 < 5, x < 5, x < y and Age < 16 are Boolean expressions.

* The comparison 3 < 5 will always give the result *true*, because 3 is always less than 5.
* The comparison x < 5 will give the result *true* when the variable x contains a number less than 5, and *false* when it contains any number that is greater than or equal to 5. The variable x must contain a number for this comparison to make sense.
* The comparison x < y will give the result *true* when the variable x contains a value that is 'less than' the value contained by the variable y. Note that in some programming languages the 'less than' operation is only defined for numbers, but in others it is defined for other data types as well.

Computer programmers use Boolean values to control selection and repetition in programs. For example, in a cinema computer system Boolean could be used to program the type of tickets people should get. Here it is written in pseudocode:

Ask the user to type in their age in whole years

Store the input data in the variable Age

IF Age<16

Set EntranceFee = 500

ELSE

Set EntranceFee = 1000

**NOT, AND, OR**

**Boolean expressions** may combine two or more Boolean values using the operations NOT, AND and OR. This is used in many different algorithms.

For example, a cinema computer system could assess if people under 16 and over 65 are charged the concession rate. This is the algorithm in pseudocode:

Ask the user to type in their age in years

Store the input data in the variable Age

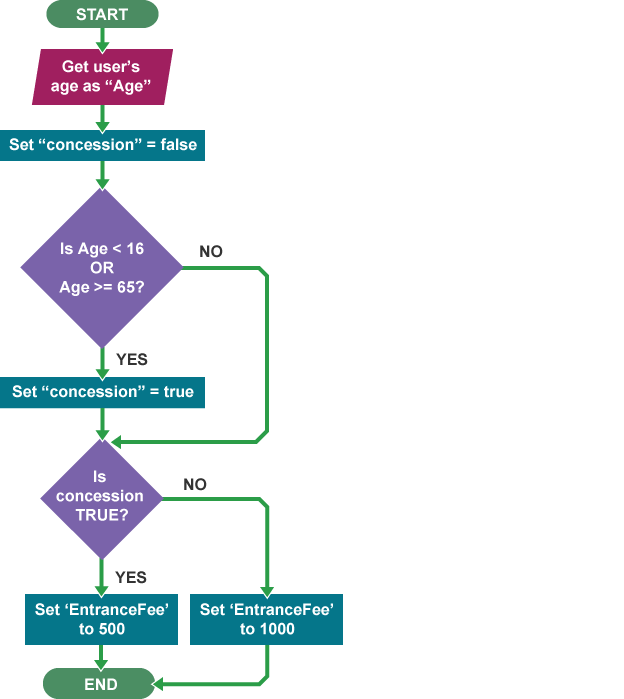
IF Age<16 OR Age>65

Set EntranceFee = 500

ELSE

Set EntranceFee = 1000

In this example, the expression Age < 16 OR Age > 65 will give the value 'true' when the user's age is less than 16 or when it is greater than or equal to 65, otherwise it will give the value 'false'.

It is also possible to create variables that will hold Boolean values for later use.

The variables in this scenario are 'Age', 'EntranceFee' and 'Concession'. 'Age' can be any integer, 'Concession' is used to hold a Boolean value and 'EntranceFee' is set according to the status of 'Concession' (500 or 1000). This can be expressed in pseudocode as follows:

IF Age<16 OR Age>65

THEN Concession

IF Concession

THEN EntranceFee = 500

IF NOT Concession

THEN EntranceFee = 1000

You can create a more specific program by creating a nested selection statement. A nested selection statement is a condition within a condition. In this case we will be nesting the if statement. We want to give both people under the age og 16 as well as people over the age of 65 a discount. However, we want those discounts to be different. Nested if statement have the purpose to make your program more specific.

IF Age<16 OR Age>65

THEN Concession

IF Age <16

THEN EntranceFee = 500

ELSE

THEN EntranceFee = 700

ELSE

THEN EntranceFee=1000

Be mindful of your indentation. Each else is indented equally as the corresponding if statement. Each if statement can only have one else statement.

# Glossary

1. **algorithm**

A sequence of logical instructions for carrying out a task. In computing, algorithms are needed to design computer programs.

1. **condition**

In computing, this is a statement or sum that is either true or false. A computation depends on whether a condition equates to true or false.

1. **flowchart**

A diagram that shows a process, made up of boxes representing steps, decision, inputs and outputs.

1. **input**

Data which is inserted into a system for processing and/or storage.

1. **instruction**

A single action that can be performed by a computer processor.

1. **iteration**

In computer programming, this is a single pass through a set of instructions.

1. **loop**

A method used in programming to repeat a set of instructions.

1. **notation**

A system of written symbols or graphics used to represent something in order to aid communication and understanding.

1. **output**

Data which is sent out of a system.

1. **program**

Sequences of instructions for a computer.

1. **programming language**

A language used by a programmer to write a piece of software. There are many programming languages.

1. **pseudocode**

Also written as pseudo-code. A method of writing up a set of instructions for a computer program using plain English. This is a good way of planning a program before coding.

1. **selection**

A decision within a computer program when the program decides to move on based on the results of an event.

1. **syntax**

Rules governing how to write statements in a programming language.

1. **algorithm**

A sequence of logical instructions for carrying out a task. In computing, algorithms are needed to design computer programs.

1. **Boolean**

A data type in computing which only has two possible values, true or false.

1. **data type**

In computer programming, data is divided up and organised according to type, eg numbers, characters and Boolean.

1. **statement**

The smallest element of a programming language which expresses an action to be carried out.

1. **variable**

A memory location within a computer program where values are stored.