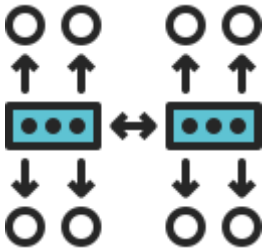
Topic 4 Theory of Computation

Algorithms



An algorithm is a series of defined steps which when followed complete a specific task. Algorithms do not contain infinite loops and must always terminate.

Pseudocode is a way of writing algorithms or other code which is not linked to a programming language. It allows programmers to communicate and share ideas without needing to all understand the same programming language.

Assignment is giving a value to a variable or constant. In pseudocode this is written as an arrow pointing to the variable or constant.

```
Count ← 77
```

Sequence is when several instructions are executed one after the other. In pseudocode instructions are written one per line, and lines are always executed in order.

Selection allows different sections of code to be executed depending on the result of a comparison. Pseudocode can use IF, ELSE IF, ELSE and END IF for selection.

Iteration, also known as loops, is the process of repeating a section of code more than once. Pseudocode can use FOR and WHILE to execute iteration. Code within the loop should be indented to make it easier to read.

Abstraction

Abstraction involves removing any details from a problem which are unnecessary and not relevant to finding a solution. It is the process of simplifying the problem down to its key details, making it easier to find a solution.

Representational abstraction focuses on removing unnecessary detail from the problem to simplify it.

Abstraction by generalisation / categorisation groups parts of the problem by common characteristics to arrive at a hierarchical relationship.

Information Hiding

Information hiding involves hiding any details or other information about an object which do not contribute to its essential characteristics. As an example, if writing an algorithm to determine how many people can fit on a bus then the people's names or hometown can be disregarded, whilst information about the height or weight of the people should be retained.

Procedural Abstraction

Procedural abstraction breaks down a complicated model into smaller parts, each of which is a reusable procedure. Abstraction is used to remove the actual values, allowing the code to be reused and a computational model formed.

Functional Abstraction

Abstracting this further can disregard the details of the procedure altogether, resulting in only a function call to provide the necessary data without needing access to or an understanding of how the function works.

Data Abstraction

Data abstraction removes details of how data is actually represented and stored, allowing new kinds of data structure to be created.

Problem Abstraction (Reduction)

Problem abstraction removes details from the problem one at a time until it is shown in a way which can be easily solved. When the problem is simplified, it often appears similar to another problem which has already been solved.

Decomposition

Decomposition divides a problem into a series of smaller subproblems, which can then be further divided. This takes a large problem and breaks it down into smaller parts which are easier to solve and manage.

Composition

Composition combines already written procedures to form a larger system when dealing with complex problems. The technique is used with abstract data types, which are formed from smaller simpler data types.

Automation

Automation used models, which are abstractions of real world problems, to solve problems. It used algorithms to form part of the code which together with these models can be executed against data structures to solve complex problems.

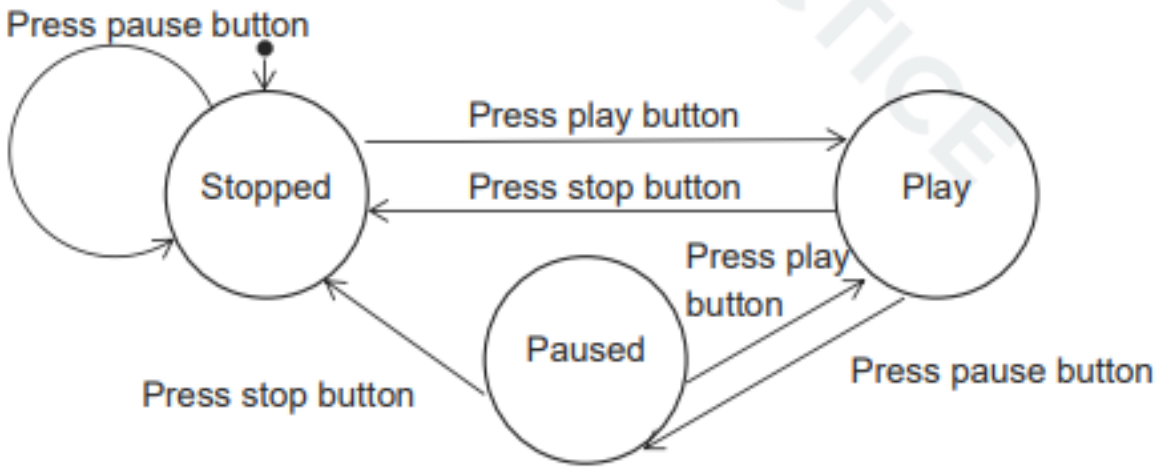
Finite State Machines (FSM)

A finite state machine is a computational model used to describe a machine which can only ever be in one of a number of finite states. The state of the machine can change based on its current state and the input data fed to it. If the input data is valid, the machine will process it and terminate in an accepting state. Transition rules define how these changes take place and how the machine should change state given certain criteria.

State Transition Diagrams

State transition diagrams are pictures which show how a finite state machine works. They are made of circles with arrows joining them together. They must always have a start state which is shown by an arrow leading into the diagram and accepting states are shown with a double circle.

The example below shows a media player, which can be in three states, stopped, play and paused. Transition functions are shown by the arrows.



State Transition Tables

Finite state machines can also be shown using state transition tables, which list the machine's current state, and how that state will change based on an input.

Current State	Input	Next State
Stopped	Play Button	Play
Play	Stop Button	Stopped
Play	Pause Button	Paused
Paused	Play Button	Play
Paused	Stop Button	Stopped