OCR Computer Science A Level

2.1.1 Thinking Abstractly

Intermediate Notes

<u>Specification:</u>

2.1.1 a)
- **The nature of abstraction**

2.1.1 b)
- **The need for abstraction**

2.1.1 c)
- **The difference between abstraction and reality**

2.1.1 d)
- **Devise an abstract model for a variety of situations**

The nature of abstraction

Abstraction is one of the most important principles in computer science. It is the act of removing excessive details to arrive at a representation of a problem that consists of only the key features. Abstraction often involves analysing what is relevant to a given scenario and simplifying a problem based on this information. This is representational abstraction.

Another form of abstraction involves grouping together similarities within a problem to identify what kind of problem it is. This is called abstraction by generalisation and allows certain problems to be categorised as being of a particular type. Thus a common solution can be used to solve these problems

Synoptic Link

You will have come across **abstract data structures**, their characteristics and functionality in **1.4.2**.

Data abstraction is a type of abstraction in which details about how data is being stored are hidden. Therefore programmers can use abstract data structures such as stacks and queues without knowing how they work.

Synoptic Link

The functions push and pop for stacks are covered in **2.3.1 Algorithms for the Main Data Structures**.

Users can also perform functions such as pushing and popping items to and from a stack without being aware of how this functionality is implemented. This is procedural abstraction. It is also used in decomposition as it models what a subroutine does without considering how. Once a procedure has been coded, it can be reused as a black-box.

Very large, complex problems make use of multiple levels of abstraction, where each level performs a different role. The highest levels of abstraction are closest to the user and are usually responsible for providing a user interface while the lowest levels interact with machine components.

The need for abstraction

At its core, abstraction allows non-experts to make use of a range of systems or models by hiding information that is too complex or irrelevant to the system's purpose. Abstraction enables for more efficient software design as programmers can focus on core elements rather than unnecessary details. This reduces the time spent on the project and prevents the program from getting unnecessarily large.

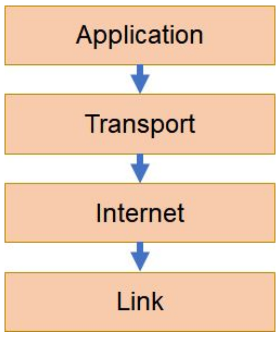
Programming languages can be separated out into a spectrum of high and low-level languages. Low-level languages such as assembly code and machine code directly interact with computer systems but are more difficult to write. Programming using machine code requires understanding binary. Assembly code requires programmers to know the mnemonics associated with the instruction set specific to each processor. High-level languages provide an abstraction for the machine code that is in fact executed when a program is run. This makes developing programs easier, as syntax in high-level languages parallels natural language and is considerably easier to learn and use. This has also made coding accessible to non-specialists.

Synoptic Link

TCP/IP is covered in the notes for 1.3.3 Networks

The TCP/IP model is an abstraction for how networks function, separated into four layers of abstraction: application, transport, internet and link.

Each layer deals with a different part of the communication process, and separating these stages out makes them simpler to understand. Each layer does not need to know how other layers work. Outgoing communication is visualised as going down these layers, while incoming information can be imagined as going up these layers.



The difference between abstraction and reality

Abstraction is a simplified representation of reality. Entities are represented as structures such as tables and databases and real-world values can be stored as variables.

Objects in object-oriented programming are also an abstraction for real-world entities. Abstraction in OOP considers the functionality, interface and properties of entities. Attributes are an abstraction for the characteristics of an object while methods are an abstraction for the actions a real-world object is able to perform.

Synoptic Link

Refer to **1.2.4** for an in-depth explanation of techniques used in **object-oriented programming.**

Devise an abstract model for a variety of situations

When devising an abstract model given a scenario, you must consider:

- What is the problem that needs to be solved by the model?

  Can the problem be solved computationally? What are the key features of the problem?

- How will the model be used?

  What sort of format does the model need to be displayed in? Consider factors such as convenience, affordability and ease of access.

- Who will the model be used by?

  How many people will be using the model? What level of expertise do they have in the subject/ discipline associated with the problem?

- Which parts of the problem are relevant based on the target audience and model's purpose?

  Remove sections that are not relevant to the problem that needs solving. Remove details that will confuse the audience.