Data Structures

One of the most important features of computers is their capacity to store, process and provide access to information or data. Computers normally have a number of devices for storing data – such as RAM (random access memory) and hard disk. Information may be stored in an ad hoc way, but, if it is, then for example it could be time-consuming to find an item. Hence, we require algorithms[1] for accessing and manipulating the information, but in order to do this efficiently it is often desirable to store data in a structured form. When we are using a computer to store information we must consider the data structure that we use to store the data, alongside the algorithms for accessing and manipulating the data structure.

Data structures also occupy space in the computer's memory. Hence for any given data set, we need to select appropriate data structures and algorithms such that the data structures fit in the available memory and the data structure is chosen so that the algorithms (that typically work on the data) are efficient in terms of processing time.

With small data sets, the memory required to store the information, or the time taken to run the associated algorithms, is often not going to be an issue. It is usually when the data sets become large that we need to study the data structures and algorithms closely; we need to be able to predict that the data structures can be managed within the memory resources of the computer it is expected to be stored on, and the algorithms can run in an acceptable amount of time.

At one level information can be thought to be stored in the primitive data types supported by the development environment (eg the programming language). Typically the environment supports integers (e.g. "int" in java or c++ (for which 4 bytes are allocated)), characters (e.g. "char" in java or c (2 bytes in java, 1 byte in c++)), decimal numbers (e.g. "float" in java and c++, 4 bytes), logical variables (e.g. "boolean" in java and c++ (theoretically 1 bit, but 1 byte for convenience in java and c++))> depending on their purpose, environments will usually support a number of other primitive data types such as a wider range of integers using 8 bytes and a higher precision decimal representation using 8 bytes.

Obviously the primitive data types use minute amounts of computer memory. However, it is when they are used as components of data structures that the memory usage becomes more substantial. For example an array data structure represents a sequence of components. Some programming languages bundle together a set of primitive data types to store a specific set of data. For example if we wanted to store a person's details then we may use characters to store their name, integers to store the date of birth. Earlier languages such as Pascal and C used "RECORD" and "struct": more modern languages like C++ or Java extend this idea further with their implementation of objects.

[1] Algorithms