Teaching guide: Data types

This resource will help with understanding data types. It supports Section 3.2.1 of our current GCSE Computer Science specification (8520). The guide is designed to address the following learning outcomes:

- Recognise fundamental types in programming.
- Relate values to their types.
- Derive the type from an expression.

Basic types

Programming is fundamentally concerned with taking data and doing something with it. It is a good ambition to have programs to process data as quickly and safely as possible. To help in this, data can be put into different types or categories.

The basic types of data found in most programming languages are:

- Integer (whole number both positive or negative)
- Real (any number that can be found on the number line) - Real number is the mathematical term, in many programming languages real numbers have cryptic names like float and double
- Boolean (only two values either True or False)
- Character (a symbol such as the letter 'a')

A character is a symbol and not a value. For example, the integer may be `14` but in terms of characters it is the symbol `'1'` followed by the symbol `'4'`. It is important to recognise this difference when writing code, to avoid the situation when the code tries to add 1 to a character and not 1 to an integer – in some languages this will crash your program, in other languages it might produce results very different to what you intended.

It is sometimes common to have a series of characters and so there is a further basic data type:

- String (a series of characters, e.g. 'cup' or '1 metre') - In Python, characters can be thought of as strings made up of one symbol.

Values

All values in a computer program have a type. Sometimes this is immediately obvious, for example:

- The value `43` has the type integer
- The value `False` has the type Boolean
- The value `'Chapter 1'` has the type string

Sometimes it is necessary to evaluate an expression to find out its type. Evaluating means computing the expression until a final value is reached and you can go no further. For example, `(4 + 6) ÷ 5` firstly evaluates to `10 ÷ 5` which in turn evaluates to the final value `2`.

- `43 ÷ 4` evaluates to `10.75` which has the type real
- The length of the string `'Chapter 1'` evaluates to `9` which has the type integer (don't forget to include the space character)
- `(3 - 2) ≥ 0` is an expression about integers but it has the value `True` and so the whole expression has the type Boolean

It is an important skill in programming to be able to evaluate expressions and recognise the type of the value.

Date and time

Data types such as integer and Boolean are treated universally across programming languages: understanding how to use integers in Python then using them in, for instance, C# and Java will not prove difficult. Some more complex data types do differ in their implementation among different languages, an example is the storage of dates and times.

Most languages have the ability to store calendar dates and times but the way that they do this varies greatly. A further complexity is the way different cultures and countries record dates. For example, the date 13 April 2015 would be written as 13/04/15 in the UK and 04/13/15 in the USA. Both of these examples share the Gregorian calendar but this is by no means the only yearly calendar system in the world – the Thai solar calendar starts counting its years using the Buddhist era which is 543 years earlier than the Gregorian calendar, so 2015 AD would be 2558 BE. Time is also relative to the time zone the user is in (and whether or not that area has daylight saving) – 1am in Bangkok could be 6pm of the previous day in London.

Computers solve this using a system time to count how many seconds (or milliseconds, microseconds or other unit of time) have passed since a commonly agreed point in time. This is then converted into the appropriate time and date format for the computer application or user. A well-known example of this is the Unix operating system that counts the seconds that have passed since midnight Co-ordinated Universal time (UTC) on the 1st January 1970, known as the Unix Epoch. This way, a particular time is recorded as an integer and then converted into whatever representation is appropriate. For example, 9:40:16 GMT on 5 July 2015 is recorded as the integer `1436089216` because that particular time is 1,436,089,216 seconds after the Unix Epoch.