Accredited

AS and A LEVEL

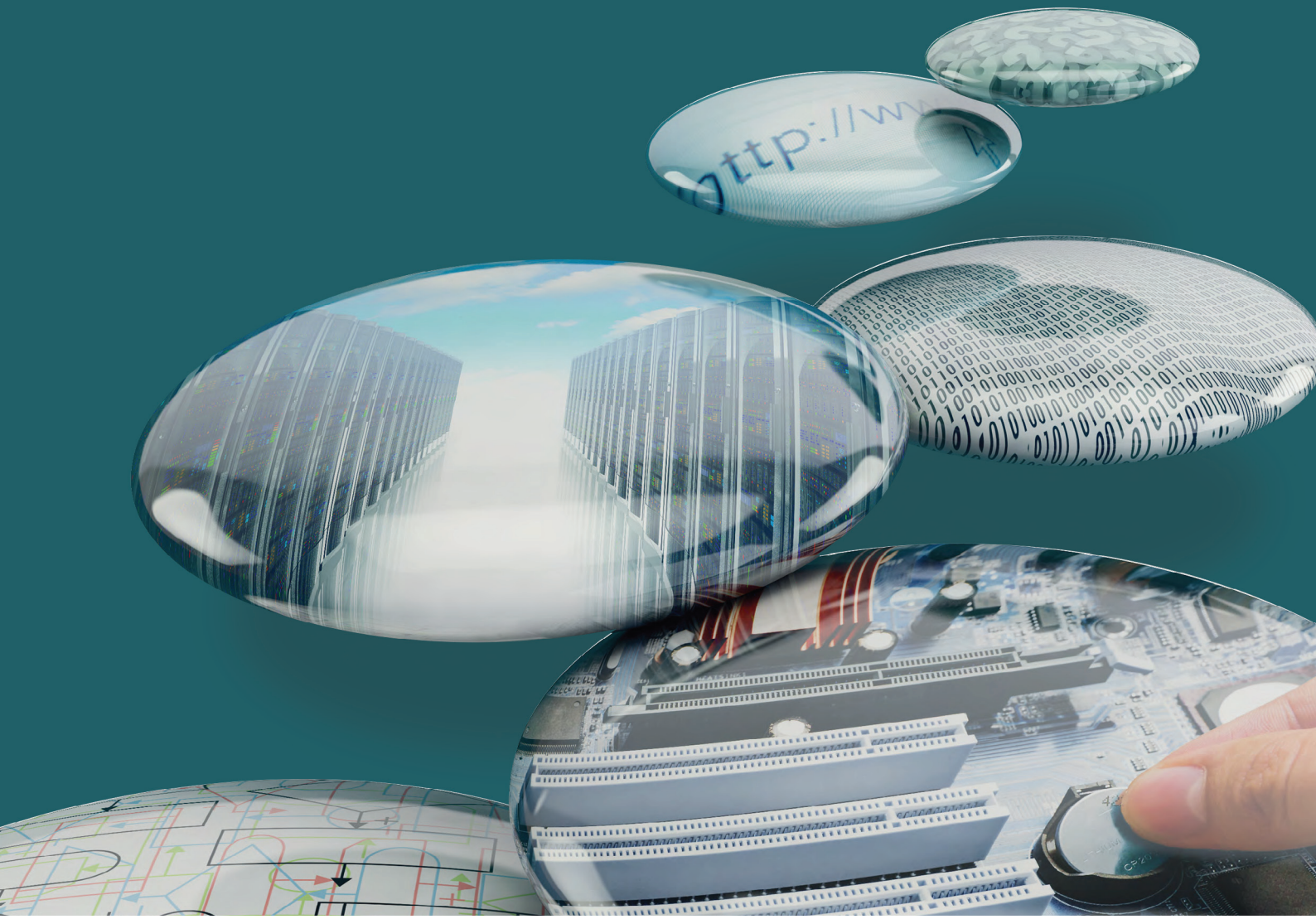
Topic Exploration Pack

H046/H446

COMPUTER SCIENCE

Theme: Data types

September 2015



CAMBRIDGE UNIVERSITY PRESS
www.cambridge.org

OCR
Oxford Cambridge and RSA

We will inform centres about any changes to the specification. We will also publish changes on our website. The latest version of our specification will always be the one on our website (www.ocr.org.uk) and this may differ from printed versions.

Oxford Cambridge and RSA Examinations is a Company Limited by Guarantee.
Registered in England. Registered company number 3484466.

Registered office: 1 Hills Road
Cambridge
CB1 2EU

OCR is an exempt charity.

Contents

This Topic Exploration Pack should accompany the OCR resource ‘Data Types’ learner activities, which you can download from the OCR website.

This activity offers an opportunity for English skills development.

Data Types

Primitive data types (integer, real/floating point, character, string and Boolean)

Primitive data types are the most basic building blocks of programming and form the basis for many more complex data structures.

Data types

	Example	What Is It?	Size
Integer	56	Represents a whole number	1–8 bytes
Real/floating point	66.45	Represents a decimal number	4–8 bytes
Character	'a'	A single digit, letter or special character	1 byte
String	'Apples'	A string is a sequence of characters	Depends how much can be stored in memory
Boolean	False	True or false, yes or no	1 bit

Primitive data types are amongst the easiest and most relatable concepts to get over to students and it is probably the very first thing that you might choose to teach. If a student has already studied programming of any sort, these should already be familiar to them. If not, a simple introduction within a programming language using conditional statements and, say, having some inputs from a user and then using these to calculate the perimeter, area and volume of a room with a string formatted user-friendly output should cover all of these concepts with no problem.

More information can be found at http://www.bbc.co.uk/education/guides/zc6s4wx/revision/2

Activity 1

You may have students read the section on the provided link and complete the provided quiz.

Representing positive numbers in binary

Representing positive numbers in binary is the typical first foray into the world of binary and understanding how computers store and deal with numbers.

Students will at first find this slightly difficult, but it is a small learning curve as once they have carried out a few examples, then the concept should sit with them.

It is the sort of thing that may be worth doing as a lesson starter (give students a few numbers on the whiteboard that they have to convert for 5 minutes) or plenary a few lessons a week to help them remember and also to help engage their brains back into thinking about binary.

There are some good activities at: http://csunplugged.org/binary-numbers

Use of Sign and Magnitude and Two's Complement to represent negative numbers in binary

The natural progression on from positive numbers in binary is to consider what happens with negative numbers, so this is a good section to cover next.

You should start off by talking about Sign and Magnitude, which uses the Most Significant Bit (MSB – the one to the far left of a binary number, e.g. for 10000000, the 1 is the MSB) to be 0 for positive values, and 1 for negative values. We do not stop here however, as this method produces inefficiencies (see table below).

After students have understood this, you might cover One's Complement, which also produces inefficiencies (see table below).

Efficiency table

For simplicity, using a 3-bit system, we can expect 2^3= 8 possible values (0–7).

Binary	Decimal (Unsigned 'Positive Values')	Sign and Magnitude (+ve First Digit is 0, –ve First Digit Is 1)	One's Complement (Flip the Bits – e.g. 001 is +1 and 110 is –1)	Two's Complement (Flip the Bits and Add 1)
000	0	0	0	0 (flipping these bits and adding 1 produces a carry bit '1000' so we ignore this and are left with '000' = 0)
001	1	1	1	1
010	2	2	2	2
011	3	3	3	3
100	4	**–0**	–3	–4
101	5	–1	–2	–3
110	6	–2	–1	–2
111	7	–3	**–0**	–1

Explanation

Sign and Magnitude – notice there are two zeroes (+ve and –ve), which is inefficient. It can be difficult to represent in hardware compared to Two's Complement, since in Two's Complement, if we wanted to add two numbers together all we have to do is add the binary numbers together, which can be achieved by a 'binary adder'. This is not the case with Sign and Magnitude or One's Complement.

One's Complement – also two zeroes and so is inefficient. Implementation issues arise in hardware also.

Two's Complement – notice how there are no problems with double zeroes. It is a lot easier to add and subtract numbers in hardware using this method, as instead of subtracting binary numbers from each other, you can simply add the numbers together. For instance, 3 – 1 = 2, but we can do 3 +(–1), which is 011 + 111. Adding these together gives us 1010, but we are using a 3-bit system so we omit the carry bit and are left with 010, which is 2, and that proves that this system works.

Activity 2

Get students to try out these different methodologies for an 8-bit number. Do they still produce inefficiencies? Do they notice a pattern for the highest and lowest negative and positive numbers that can be represented using these methodologies?

Two's Complement is simply inverting each bit for a positive number and then adding 1. You may need to go through how to add 1 to a binary number first however, such as how to carry bits.

This is a very good video from Dave Collins that shows how you might teach these concepts to students on a whiteboard (parts 1 and 2): https://www.youtube.com/watch?v=9W67I2zzAfo

Activity 3

Have students study this section of the BBC Bitesize website about representing numbers in binary and then self-test using the provided activity: http://www.bbc.co.uk/education/guides/zjfgjxs/revision/5

Addition and subtraction of binary integers

The prospect of adding binary numbers could strike fear into many students but it is easier than adding denary numbers since the carry bit can only be 1 and there are only four possible calculations (0 + 0 = 0, 0 + 1 = 1, 1 + 0 = 1, 1 + 1= 0 carry 1)

Example

Binary addition may require carrying over values if there are two 1's in the same column.

If we want to find the sum of 0010 (2) + 0111 (7) = 9

Then

0 0 1 0

0_1 1_1 1 1

---

1 0 0 1

See that there is a clash when adding two 1's in the same column. In binary 1 + 1 = 10, so the output is 0 with 1 carried over.
A good guide to binary addition can be found at:
http://www.bbc.co.uk/education/guides/zjfqjxs/revision/1

Subtraction

If a number is represented as Two's Complement, you simply 'add' to carry out the subtraction, that is 3 – 5 is done by adding 3+(–5).

Represent positive integers in hexadecimal

We refer to binary as a 'base 2' system because there are two possible values for each digit. Denary is base 10 because there are 10 possible values for each digit. Hexadecimal is base 16. For convenience, engineers used to group three binary digits together, for example 111, which could achieve up to eigth possible values. However, as computers got bigger it was more convenient to group together four numbers, for example 1111, which could represent up to 16 possible values. Since there are no symbols that can represent 11, 12. etc., A to F are used after 9.

Again, representing positive integers as hexadecimal is relatively straightforward, as each hex digit represents a 4-bit binary sequence (otherwise called a 'nibble' as opposed to 8 digits, which we call a 'byte').

Denary	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Therefore, 1001 1111 = 9F. You just have to split the 8-bit number into two 4-bit numbers ('nibbles') and convert each 4-bit number into its hex value, then glue them back together.

A simple rundown can be found at: http://www.bbc.co.uk/education/guides/zp73wmn/revision/1

Convert positive integers between binary, hexadecimal and denary

In either way students usually find it conceptually simpler to go from denary to binary to hexadecimal and the other way around, each time using binary as the intermediary.

A good resource can be found at: http://www.bbc.co.uk/education/guides/zp73wmn/revision/3

Activity 4

Give the students a list of 8-bit binary numbers on the board. They have to figure out what the denary and hexadecimal equivalents are. Use the table above to help both yourself and students who are struggling.

Representation and normalisation of floating point numbers in binary

Students who have trouble with mathematics may struggle at this point if they do not understand exponents and the idea of 'standard form/scientific notation', so some time should be dedicated to re-capping on this.

An example of what we mean by a mantissa and exponent is:

5.3×10^{-5}

5.3 is the mantissa and 10^{-5} is the exponent.

Because the exponent is –5, we move the decimal point five spaces to the left. Inversely if it was positive, we would move it to the right.

Therefore, the number this represents is 0.000053.

Wikipedia link to explain 'standard form': http://en.wikipedia.org/wiki/Scientific_notation

A particularly good guide to the representation, converting between binary and denary floating point numbers and normalisation can be found at:

http://community.computingatschool.org.uk/resources/363

It is also worth talking about trade-offs between the precision for the mantissa and the exponent, since if a larger part of the binary number is dedicated to the mantissa you can gain more accuracy, whereas if the exponent is larger, then you can represent increasingly larger/smaller numbers.

A good PowerPoint on the subject of trade-offs can be found at:
http://community.computingatschool.org.uk/resources/364

Activity 5

Have students complete the exercises from the above links.

Bitwise manipulation and masks: shifts, combining with AND, OR and XOR

Most of the time when you are a programmer you don't have to be concerned with operations at bit level (low level) as you are likely to use higher level data types such as integers or floats, which, as we have seen, are represented in memory as a series of bits. There may be certain cases, however, such as controlling LEDs from a microcontroller (such as an Arduino), encryption or graphics processing where you might want to be able to manipulate bits.

Activity 6

There is a particularly good video from the 2008 Royal Institution Christmas Lectures on XOR (exclusive OR) encryption at: http://youtu.be/LGGFpQMOAYQ?t=7m24s

In addition, bitwise operators are one of the most fundamental processes that are undertaken by the computer's processor when it is doing calculations.

An example of a bitwise manipulation is AND, where we use the idea of an AND gate where we multiply each digit (i.e. 1 x 1= 1, 1 x 0 = 0), for example:

1 1 0 0
AND
1 0 1 1
---
1 0 0 0

In practice, bitwise manipulation is relatively straightforward and relates well to the part of the specification dealing with logic gates, although both can easily be taught independently of each other.

A good explanation can be found here: http://en.wikipedia.org/wiki/Bitwise_operation

The below PowerPoint also has a good explanation complete with student exercises and examples of how bitwise manipulation is used in practical terms: http://community.computingatschool.org.uk/resources/145

Activity 7

Have students complete the homework from the above link.

How character sets (ASCII and UNICODE) are used to represent text

Character sets, again, are relatively straightforward and are interesting for students since it relates to what happens when they use the keyboard.

ASCII is a 7-bit binary code that allows for 128 characters, which includes lowercase, uppercase and special characters. Extended ASCII is an 8-bit code that represents 256 characters. Unicode uses between 8 and 32 bits per character meaning it can represent characters in all languages. It might take up more room when saving documents, but if a company operates in many different languages they would most likely use Unicode.

A good explanation can be found here:
http://www.bbc.co.uk/education/guides/zp73wmn/revision/4

Teacher preparation

In terms of teacher preparation the best thing to do would be to firstly have a good grasp of these specific topics:

- Primitive data types and being able to use them confidently to create conditional statements in the programming language you are most likely to use, although a variety of languages is preferable.
- Representing positive numbers in binary, and being able to complete the reverse.
- Use of Sign and Magnitude and Two's Complement to represent negative numbers. It would also be useful to understand One's Complement and why Two's Complement is the most useful.
- Addition and subtraction of binary integers.
- Representing positive numbers in hexadecimal.
- Now that you have covered hexadecimal, it would make sense to have practice on converting positive numbers between denary, binary and hexadecimal.
- Representation and normalisation of floating point numbers in binary. This is probably the trickiest bit.
- Bitwise manipulation and masks (AND, OR and XOR).
- How ASCII and UNICODE are used to represent text.

You will undoubtedly pick up various methodologies that make sense to you of how to approach various problems, and will be able to pass these on in your teaching to students.

It is then a case of getting sufficient worksheets prepared with examples and enough variety to stretch the most able students. Some of the links provided will also have additional exercises to be able to worth through.

Student Sheets

Activity 1: Cup binary machine

Students should enjoy making this device. All that is needed are nine coffee cups and a marker pen and students enjoy the novelty of doing something a bit different. Confidence levels should rise quickly with converting denary to binary and vice versa. This is also a good activity for weaker students who may be initially more dependent on the device.

Activity 2: Binary addition and Two's Complement worksheet

This worksheet is an example of how easy it is to create chances to practice the key skills within this topic.

There are tools available online and also Calculator in Microsoft Windows, which could let students check their answers (conversion from binary to denary and the reverse). Other tools are available online for Two's Complement and floating point numbers, amongst others.

Activity 3: Floating point worksheet

Again, this worksheet contains example questions with worked through solutions. You could get students to design their own if they are finished or go online to explore more on the topic.


We'd like to know your view on the resources we produce. By clicking on the 'Like' or 'Dislike' button you can help us to ensure that our resources work for you. When the email template pops up please add additional comments if you wish and then just click 'Send'. Thank you.

If you do not currently offer this OCR qualification but would like to do so, please complete the Expression of Interest Form which can be found here: www.ocr.org.uk/expression-of-interest

OCR customer contact centre
General qualifications
Telephone 01223 553998
Facsimile 01223 552627
Email general.qualifications@ocr.org.uk