Natural numbers

1 Introducing the natural numbers

The *natural numbers* are 0, 1, 2, 3, ...

The set of all natural numbers is written $\mathbb{N}$. It is an infinite set.

2 What are natural numbers used for?

Natural numbers have two basic purposes.

One purpose is to answer the question "How many?". For example: "There are 3 people and 0 elephants in the room." (Here we use a natural number as a "cardinal".)

The other purpose is to answer the question "Which one?" For example, say John is at the front of the queue, followed by Mary, then Pat, then Wilf, then Xerxes. What position in the queue is Wilf? (Here we want to use a natural number as an "ordinal" or "index".)

In order to answer this question, we need a convention. We can say that Wilf is the *exclusive third* person in the queue. This means there are three people up to and excluding him, viz. John, Mary and Pat. Or we can say that Wilf is the *inclusive fourth* person. This means there are four people up to and including him, viz. John, Mary, Pat and Wilf.

The inclusive convention is also called "indexing from one". The exclusive convention is also called "indexing from zero".

English uses the inclusive convention by default, so we would just say "Wilf is the fourth person in the queue". But the exclusive convention has some advantages and is often used in computer science and maths. For example, array indexing in Java.

It is *essential* to be clear about which convention you are using in any given context. In programming, one can very easily make "off-by-one" errors, and careful thinking is needed to avoid this.

3 Representing natural numbers on paper

Each natural number n has a *successor*, written $\mathsf{S}n$. We can express any natural number using the successor operation and 0. For example, the number 5 would be written $\mathsf{SSSSS}0$. This is called *unary* notation. Very natural from a theoretical and conceptual viewpoint, but horribly inefficient!

Instead we use *positional notation*, historically called Arabic numerals (with the zero coming from India). For example, we can use base ten, also called *decimal* (or denary). Then we have ten digits 0,1,2,3,4,5,6,7,8,9. For example:

$$\begin{aligned} 4389 &= 9 + 10 \times (8 + 10 \times (3 + 10 \times 4) \\ &= 9 \times 10^0 + 8 \times 10^1 + 3 \times 10^2 + 4 \times 10^3 \end{aligned}$$

It's sometimes useful to include leading 0's. For example the above number can be written 00004389, if you just happen to want 8 digits.

Although ten is the most popular base, because humans generally have ten fingers (not to mention ten toes), any other natural number greater than 1 can be used. For computer science, base two, also called *binary* is very important. This means that we have two digits 0 and 1. For example

$$\begin{aligned} 1011_2 &= 1 + 2 \times (1 + 2 \times (0 + 2 \times 1)) \\ &= 1 \times 2^0 + 1 \times 2^1 + 0 \times 2^2 + 1 \times 2^3 \\ &= 1 + 2 + 8 = 11 \end{aligned}$$

The subscript 2 indicates that we're using base 2. If there's no subscript and no other indication of the base, then assume that the base is ten.

A binary digit is called a *bit*. A group of eight bits in computer memory is called a *byte*.

Sometimes you'll encounter base 8, also called *octal*, where the digits available are 0,1,2,3,4,5,6,7. And sometimes you'll encounter base 16, also called *hexadecimal* (or hex), where the digits available are 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F. Here we have A $= 10$ and B $= 11$ and C $= 12$ and D $= 13$ and E $= 14$ and F $= 15$. These bases are useful because an octal digit corresponds to a group of three bits, and a hexadecimal digit to a group of four bits (also called a nibble).

4 Representing natural numbers on a computer

In the case of Java, there is no type for natural numbers. (There is for integers, but we'll come to this later.) But the C language does have one, called `unsigned int`. The problem is that only 4 bytes (=32 bits) are allocated. So a variable of this type can go all the way up to $2^{32} - 1$, which in binary is as follows:

$$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111$$

When we add 1, we run out of space and get

$$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000$$

So a programmer in this language has to be careful! The "natural number type" is not a genuine natural number type, although it's good enough for many applications.

5 Operations on natural numbers

Apart from successor, there are many operations on natural numbers.

- Addition, e.g. $3 + 4 = 7$.
- Multiplication, e.g. $2 \times 7 = 14$.
- Exponentiation, e.g. $2^3 = 8$ or (my favourite) $0^0 = 1$.

Then we have some inverse operations. For example, subtraction, e.g. $5 - 3 = 2$, but only if the number being subtracted is less than or equal to the original. This is a limitation of natural numbers.

Another example is `div` (which divides as many times as possible) and `mod` (which gives the remainder). For example:

$$\begin{aligned} 432\ \texttt{div}\ 100 &= 4 \\ 432\ \texttt{mod}\ 100 &= 32 \end{aligned}$$

Let's put this more abstractly. Say that b is a nonzero natural number and a is any natural number. Then $a = m \times b + r$ for natural numbers m and r, with $r < b$. For example, $432 = 4 \times 100 + 32$.

You have learnt algorithms for arithmetical operations in base ten, but they are easily adapted to other bases. For example, you should be able to add or multiply in binary.

6 Equational laws

Let's just look at addition and multiplication. They satisfy many laws.

- Addition is commutative: $a + b = b + a$, so the order doesn't matter. It is associative: $a + (b + c) = (a + b) + c$, so we don't need to write brackets. It has a neutral element 0, meaning that $a + 0 = a$.
- Multiplication is commutative: $a \times b = b \times a$, so the order doesn't matter. It is associative: $a \times (b \times c) = (a \times b) \times c$, so we don't need to write brackets. It has a neutral element 1, meaning that $a \times 1 = a$.
- Multiplication distributes over addition: $a \times (b + c) = a \times b + a \times c$. It has an annihilating element 0, meaning that $a \times 0 = 0$.

The above laws are called the *commutative semiring laws*, and we can derive other laws from them, such as $(a + b) \times c = a \times c + b \times c$. I'm not suggesting that this law is less obvious than the ones above, but it's interesting to see how we can derive mathematical facts from a small group of basic "axioms".