**Binary Representation**

Internally, computers store data in memory (i.e. numerical values, characters) using numerical encodings. That is, everything in computer memory is stored as a number.

Of course, since those numbers can represent different data ***types***, we need different ways to encode those different types of data. The encoding method is different depending on whether the values to store are integers, double, characters, etc.

*Example*: *Storing characters*

The numerical codes used to represent characters are well established. ASCII tables, for example, list the ASCII numerical code that correspond to any given character. These tables can be easily found on the Web (https://www.asciitable.com/)

The programming language we use (C++ in our case) will take care of encoding the character values used in our programs so they are stored in memory using their corresponding numerical value as it appears in the tables. If we instruct our program to display a character, the numerical value stored in its memory location will be fetched and the inverse process will be carried out: the numerical value will be decoded and the corresponding character will be displayed.

For example, the numerical value corresponding in the ASCII to the capital C is 67. If we instruct our program to store the character 'C', the actual data saved to memory will be 67.

If, later on, our program wants to display the character stored in that memory location, the number stored there (67) will be retrieved and automatically translated into the character 'C', which will be displayed.

*Bits and binary*

We know that values of different types are all encoded and stored as numerical values. But how are those internal values represented? Since computer memory is built as a set of bits the are ON (which represents the number 1) or OFF (which represents 0), it can only represent values as a combination of 0's and 1's. This is called the binary number system.

For example, we know that 67 is the numerical ASCII code for ´C´(capital c). Therefore, when storing the character ´C´in memory, number 67 would be stored instead. But 67 cannot be stored directly, as computers can only store numbers as sequences of 0`s and 1´s. therefore, 67 will have to be converted to the sequence of 0´s and 1´s equivalent to it. That is, it will be converted to its binary representation.

1000011

How can we verify that this binary number is equivalent to 67?

Any whole number represented in binary form can be converted to the decimal number system (our usual number representation) by multiplying the binary number by growing powers of 2, starting from right to left.

In our case: $1 + 1x\ 2 + 0\ x\ 2^2 + 0\ x\ 2^3 + 0\ x\ 2^4 + 0\ x\ 2^5 + 1\ x\ 2^6$ = 67

Following similar steps, we see that the binary number 11010 is equivalent to

$0 + 1\ x\ 2 + 0\ x\ 2^2 + 1\ x\ 2^3 + 1\ x\ 2^4$ = 26

*More examples*

1011011 is the binary representation of 91

10 is the binary representation of 2

11110 is the binary representation of 30

*Additional resources*

There are many Web videos explaining the steps needed to convert binary numbers to their decimal system representation. For example https://www.youtube.com/watch?v=VLflTjd3lWA. You can find more by going to the YouTube page, for example, and searching for "binary to decimal."