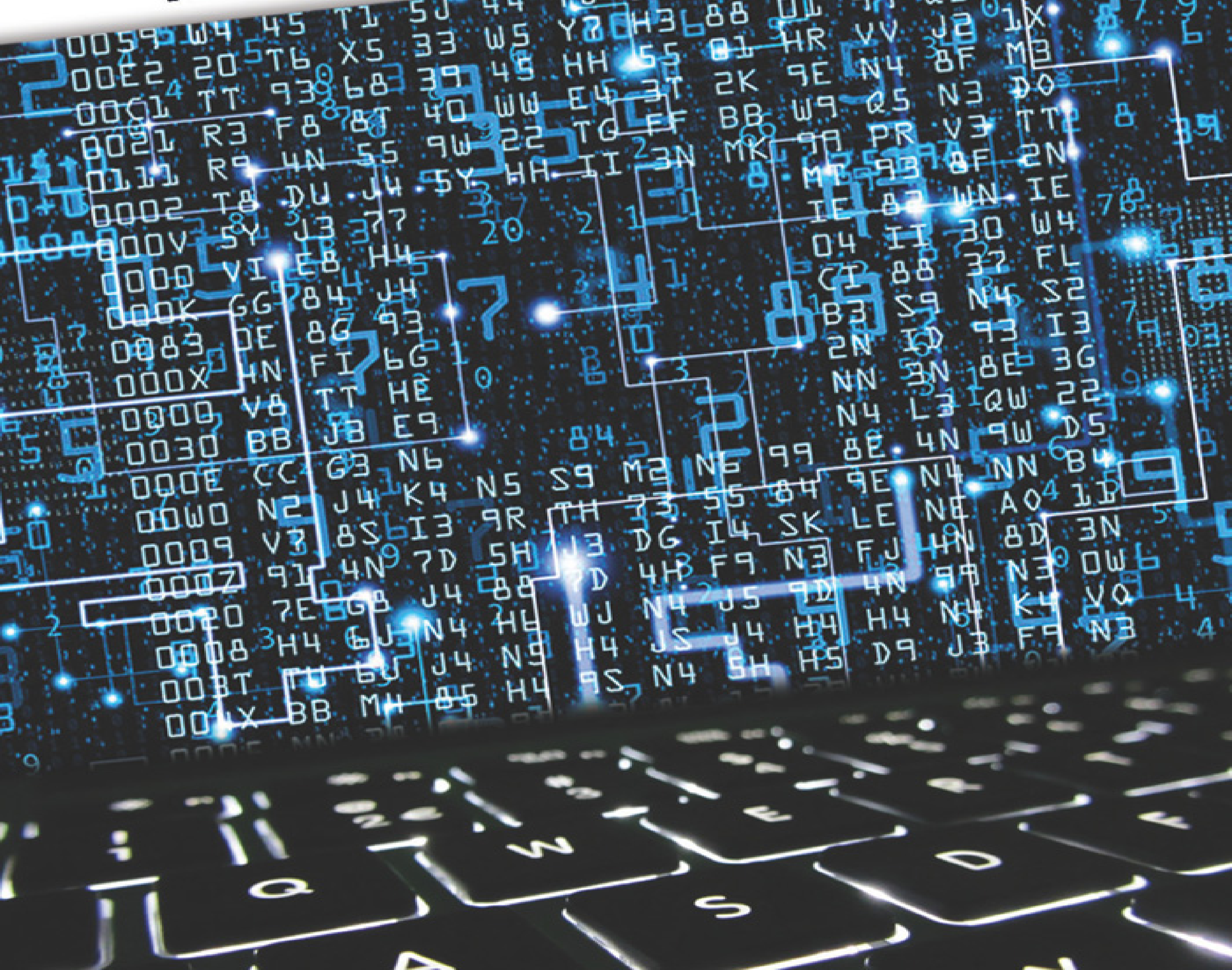
WHY CODE?

KRISHNAV BAJORIA



Published in India by Prowess Publishing,
YRK Towers, Thadikara Swamy Koil St, Alandur,
Chennai, Tamil Nadu 600016

ISBN: 978-81-944634-7-4
eISBN: 978-81-944634-9-8

Library of Congress Cataloging in Publication

Today the whole world has become some sort of a global village where every person is sharing their thoughts, ideas and feelings via the internet through chatting, gaming and many other related activities. In this world of today it has become quite important to understand the working of such systems. This understanding is brought through computer science and one of the important disciplines of this field is computer programming. Today in this era where the technology is advancing towards a new level altogether with robots and machines using their 'artificial intelligence' are taking the whole world with a surprise. In the last decade artificial intelligence has gone from a science fiction dream to a critical part of our life. We use AI systems to interact with our phone through Siri and Alexa, cars like Tesla interpret and analyse our surroundings to intelligently drive themselves, Amazon monitors our browsing habits and intelligently serves our products it thinks we are likely to buy, even Google gives us our search results based on who it thinks we are. But if we ask ourselves the question on what basis does such work happen? This is where computer programming and algorithms come into play. Now the question may arise on what platform do we program and if how does the computer understand it? Of course it doesn't understand English nor other regional languages. So through which medium are we communicating with our computer? … It is … through a bunch of high-level languages. So the next question that may arise is how does a high level language be understood by a computer? The answer to this question is that there is a certain device called a compiler which actually converts high level languages to low level languages which are understood by the computers which are actually in the form of 0's and 1's. Then there is another question which is "why don't we just write the low level language"? and the answer to that is very simple as it becomes more complicated to write low level language. For example if you want to give a number for some arithmetical operation like 27 the computer doesn't understand 27 instead it understands the binary code for this which is 11011. Now the question may arise how did we get this? How can a human formulate such a way to get such numbers to machine readable language? It is programming and the logical reasoning behind such algorithms that gets us to the answer.

When human being acquired language we just did not learn how to listen but how to speak, when we acquired text we just did not learn to read but also to write and now since we have acquired computers we should not just learn how to use them but to program them. Nowadays people use a lot of social networking websites without actually knowing the code or the purpose of the software. For example if you ask a child say, what facebook is used for and he or she will tell you that facebook is here to help me make new friends which is not the case. They are looking to monetize people's relationships. Now if you don't know what the software your using is for, then you are not using it but being used by it. In any age whoever holds the key to programming ends up building the reality in which we live today.

There is a key feature in programming which is known as abstraction and as the word suggests it is used for hiding background details and only displaying the essential features and this art of programming has influenced a lot of consumers and the gap between the users that are the people who use the software and the commands which are used to execute those softwares are being known by only a few. Nowadays we use this word a lot of times which is called as an "interface" like the GUI, the graphical user interface or any other interface it may be, it is also another method under which programming is carried out. These also promote this feature of abstraction. Giving a real life example, the driver who drives the car doesn't need to know the internals of the system inside the car but just the handles and breaks and a few buttons. Now we can imagine how intricate a cars internal structure would look like. So we can understand that this art has been worked upon and modified to a great extent. If you want to build such a device which is very intricate and show it to the world as amazing as a car you could just use the help of programming and the algorithms that are meant to design the system and you are done. (You could do anything in this world that you have dreamt of in your dreams like creating your own game or your very own startup of a particular technology just by learning how to program). Who knows you could be one of the next great thinkers and programmers after Steve Jobs or Bill Gates? Both had their own vision and their own interest in programming and it was because of this interest and their skill and vision that they reached their goals and we all know them for their own inventions and their impact on their world even today.

A very similar comparison of a programmer is of a painter. How? Just like painters paint through their brushes and colours to make their picture come alive similarly a programmer designs his code in such a way that it becomes very useful and very comfortable for the user to interact with.

The painter is the creator of the art which is completely his or her piece of work similarly a programmer who writes code is the creator of that piece of code whose credit goes completely to the programmer. It is said ‘a work of art is magical, and a piece of code is legendary’. A programmer can be thought at one level as not just one who creates or writes code but also creates and discovers new and effective solutions to problems whose difficulty range can vary from very low to very high and complicated problems. To give an example it could be from helping a user to solve five calculations of whose input the user gives and of whose output the programmer has to give, to a problem say which is a very well known problem called as the traveling salesman problem where the user has to go through a set of points within minimum time or help him devise a method to take the shortest route possible through the set of all points given. The previous was an example of an approximate algorithm problem and they could be of various types where the solution to a particular problem may be efficient or not which we denote it as P or NP problems where there are seemingly no possible efficient solutions to a problem. To solve such problems which concerns us in such a major way is an honour for every programmer.

Painters paint the visible through vision that is through their imagination while programmers paint the invisible through their foresight in code.