Clark Fisher
998172053
Winter 2013

Abstractness in Computer Science

Consider a map outlining a busy subway system in a very crowded city such as London or New York. You would see an image represented by an organized body of straight lines and dots that seem to miraculously bind themselves together in a coincidentally neat manner. However, if you take a second map detailing the actual construction of the tunnels of these subway systems, you will notice that in reality these tunnels conduct themselves in a very opposite behavior. There are anything but a straight line, and are rather a combination of very uniquely shaped trails that happen to attach at a given train station. Why is there such a drastic difference between these two maps? The answer is because it follows the principles of abstractness. Understand that you could probably make your way from one side of a given city to another using the second map, but would you have any difficulties doing so? The second map is not simplistic or easy to read, and potentially mistakes can be made when interpreting it due to the amount of information that you would have to process in your mind. The idea that anything can be simplified by molding it to it's most basic and base form makes subway maps easier to read, but it also makes writing algorithms in computer science simple and comprehensive.

Take for example the recursion principle in computer science. If you are finding and computing the factorial of 10 [10!], there are two ways to go among that. You could possibly write out using recursion that 10!=n*9!, 9!=n*8!, 8!=n*7!, and so forth until you get to the factorial of 1, but could that possibly have been written

faster using abstractness. What if you could program an algorithm that computed the above recursion relation for you so that you did not have to write out every step of the equation? What if the program could take any factorial and know that it needs to take n-1!*n and add it all together. Assuming that it is true, you could reduce the amount of lines that you would need to write a program to solve the recursion relation enormously. This is abstraction, or the idea of simplifying lines of code using cleverly constructed yet simpler lines of code. Basically, by thinking outside of the box and breaking down every component of a program to its most basic form, computer science programmers are capable of uniformly carrying out commands that would normally take up too much time to write or be too complicated to keep track of.

However, why do we need to incorporate abstractness into computer science? At our Freshmen Seminar's current level of understanding the fundamental principles of programming, we may not fully appreciate what a lasting impact it can do for individuals writing code, especially when you consider programs that are hundreds if not thousands of lines long. Without abstractness, the process of writing programs would be so long and tedious that it would make every computer science task performed seem extremely difficult and deter individuals new to the subject that would like to give it a shot. Ultimately, abstractness is a programmer's ability to evolve with time. As we dive further and further into the 21st century, we are witnessing an influx of technology that is far more superior and advanced than its ancestors. Take the cell phone for example. Originally home phones were established, which later found its way into our pockets in the form of a cell phone.

Now, smart phones have been developed which not only can answer and send out phone calls and text messages, but access the internet and carry out other functions much similar to a computer. Within what really is only twenty years of time passed, programmers had to transition their code from (what would be now) a simple device to send and receive calls into (what would be then) a super computer capable of sustaining an individual's personal and business life. Through the idea of abstractness, programmers are finding simpler and more efficient ways of writing the code of products and programs of the future, and all technological progress would be at a screeching halt if they were not capable of carrying out this process in a mildly stress free manner.