What is programming?

Khambar Dussaliyev

March 19, 2022

Abstract

This essay is intended to answer (very briefly) to a question: *What is programming?* Although computer programs nowadays are pretty ubiquitous in nature, they remain being *black boxes of magic* for most people, even tech-savvy ones.

However, being a *black box of magic* to some extent is a requirement for some commercial software (especially when there are trade secrets involved) natural. But this essay is not intended to answer how every software works in details, but *how every software works, in general*.

Contents

1 Introduction

Basically, *everything* you do on a computer is running some program. Everything, no matter what you do is a result of executing and working with some program. Your OS[1] *is a program.* Your internet browser is a program. Your media player, file explorer, video game, media editing software, office software *are programs. Everything that allows you to interact*[2] *with computer (and not just bare metal) is a program.*

So, essentially, *what is a program?*. According to Merriam-Webster[3] definition[4] (applicable for us), we can use following as a definition:

> program is a sequence of coded instructions that can be inserted into a mechanism (such as a computer)

The gist of it being — *sequense of instructions.* So every interaction we can possibly have with a computer is somehow just a set of instructions. But *how computers do understand out instructions*? If I just shout into the microphone some command, computer will not just do as I say[5]. The same effect will have some instruction that I carefully write them in some text document, using Microsoft Word, for example.
How do I make computer to understand what I want from it? To understand this, *we must first understand what is a computer.*

[1] Operation System - Windows, Linux-based, MacOS, Android, iOS etc.

[2] Interaction with a computer here and on will not take physical interaction with a bare metal in account

[3] Here and on Merriam-Webster dictionary is referred to as a 'general-scope' dictionary to avoid technical details redundant for this essay

[4] https://www.merriam-webster.com/dictionary/program

[5] Provided, there is no running program, responsible for such behavior

2 What is Computer?

2.1 It's all about information

Let's once again refer to Merriam-Webster dictionary for a *computer* definition[6]:

> computer is a programmable usually electronic device that can store, retrieve, and process data.

So, a computer directly tied to all sorts of *data maniplation*. But what is data, and how can it be manipulated? Data is pretty much any factual information. Weather outside a window? Data. T-Shirt you're wearing? Data. Dusty books in my shelves? Data. But there are a certain layers present. The fact that I'm wearing a T-Shirt is data (even if I'm not — also data). What color it is is also, most certainly data. Is there any text or image present? Both the existence and *information* in it — also data. Let's also not forget about colors, sizes, fonts. We are living and have always lived in an enormous ocean of data. Nowadays, with our technology even more so.

But do we have use for this data? Well, the answer is — it depends. To assess data without well-defined goal will almost always result in you drowning in said data without much progress, since world offers us practically indefinite source of it. We must put our data into some perspective, some context. Once we put our data in some context and it becomes *useful* for us, *it becomes information*.

> To somehow navigate in this world, we must put our data in context. Data with given context becomes somewhat useful. Such data called *information*.

Information is much more well defined than data. *We can measure information, actually*. We even have a science discipline, called *Information theory*, that researches all about information, from mathematical and engineering standpoint. Not only that, but we have an entire industry, called *Information Technologies* built entirely on a foundation provided by Information theory and adjacent disciplines.

But until we dwell into technical stuff, we must also remember, that *we used to operate with information*. Our brain is a natural computer, disecting data into information that we use in our everyday life. How come I am so sure to call that information and not just raw data? Well, that greatly depends on a scale, but *our brain have very defined goals*. One of the main said goals being to *keep us alive*. Therefore there is a context, and brain will always tend to categorize things (organize data) based on this goal[7], making it somewhat useful, therefore making it an *information*. Even the simple fact, that we don't notice our noses, although our eyes do see it constantly, tells us how natural brain is in working with information.

[6] https://www.merriam-webster.com/dictionary/computer

[7] If your goal at the moment being something more specific, than just stay alive, many of the information brain gives you still raw data

So, being a natural organic computer, we must first understand what we do with information, to have insight on what computers do with information. All processing of information, our minds in work is mostly an internal process. *Sooner or later there is a point, when we must exchange said information.* So, how do we exhange it? We have an enormous number of ways, actually. We can say something to another person, write it down, pass via somebody a note, we can just hint at something. We can send message in a messenger, send an email, send a radio-signal, we can knock a Morse code. *We are practically limitless with one major nuance – the other side must be able to understand us.* There is no point in sending email to a person, who can't use it[8].

We now can conclude one fundamental distinction: information itself is mostly independent from it's carrier. To demonstrate it more clearly, let us consider following example: I want to send information to my friend at the table, with the main message being *'pass me salt'*. There is definetely a context: we are at the table, eating food, there are at least two parties involved (me and my friend), and I expect salt to exist somewhere at the table. I can pass this information with a various number of ways, provided my friend understands me. Just to mention a few:

- Say to him 'pass me salt'
- Ask him to pass me salt in other language, he is familiar with
- Write a note to him 'pass me salt'
- Write a note in foreign language, he is familiar with
- Get his attention and non-verbally point at salt
- Write him a message in messenger, expecting phone to be near them
- Get his attention and use ASL or alternative, provided he is familiar with it
- Exclaim obviously 'Oh! This food will be so much better with salt! I wish somebody passed it to me now', provided he understood our hint
- Rhytmically knock with Morse code, provided he understands it

In all aforementioned examples we can clearly see, that the gist of our 'message' stayed the same. *We did pass a more or less the same information* in each and every case. Despite the medium being completely different, if our friend can undestand us, nothing really changed for him or us. In such cases the 'main message' containing an actual useful information we are willing to exchange usually colloquially called a *'payload'*. However, despite our 'payload' being virtually the same, we did pass some additional information (or data – depending on context) along the way, didn't we?

[8]In general. Sometimes we are interested only in sending information, not concerned by an actual delivery. Some legal procedures can be of an example

Let's use an above list one more time, but will provide additional few details, just for example:

- Say to him 'pass me salt'
  - In what voice tone?
  - How loud did we ask?
  - What face expressions followed along our request?
  - Was there any gesticulation involved? How intense?
  - In what speed we asked?
- Ask him to pass me salt in other language, he is familiar with
  - What language?
  - Was there any context in using this language?
  - In what voice tone?
  - How loud did we ask?
  - What face expressions followed along our request?
  - Was there any gesticulation involved? How intense?
  - In what speed we asked?
- Write a note to him 'pass me salt'
  - What font did we use?
  - Is it hand-written?
  - Font color?
  - Font size?
  - Paper type?
  - Paper size?
  - Was paper plain white, or was it with pictures?
- Write a note in foreign language, he is familiar with
  - What language?
  - Was there any context in using this language?
  - What font did we use?
  - Is it hand-written?
  - Font color?
  - Font size?
  - Paper type?
  - Paper size?

  - Was paper plain white, or was it with pictures?

- Get his attention and non-verbally point at salt
  - Were we mumbling at the same time?
  - How we got his attention? Tapped his shoulder? How strongly?
  - How did we point? With a finger, palm, node?
  - What face expressions have we used?
  - How fast did we do it?

- Write him a message in messenger, expecting phone to be near them
  - Have he seen us typing a message?
  - How fast he reacted?
  - Was there any notification
  - Did we send an emoji?
  - Did we send some attachment?
  - We sent one message or several?
  - Did he read it?

- Get his attention and use ASL or alternative, provided he is familiar with it
  - How exactly did we phrase our request? By letters or by gests?
  - How fast did we transmit?
  - We followed along with our lips?

- Exclaim obviously 'Oh! This food will be so much better with salt! I wish somebody passed it to me now', provided he understood our hint
  - What intonation did we use?
  - How loud did we say it?
  - Where to did we look?
  - Do we have any specific accents?
  - Was there an emphasis on some words?

- Rhytmically knock with Morse code, provided he understands it
  - What period of time we used as an interval?
  - Did we repeat our message? How many times?
  - On what surface did we transmit?
  - Did we use our knuckle? Spoon? Knife?

So, we can conclude, that despite our *payload* being practically the same, we did pass additional information along with it. To put it into perspective, It's somewhat similar, as if we were asked to describe an envelope, it's size, stamps on it and additional notes, ignoring the payload, being a letter inside said envelope. Such auxiliary information, which is more often than not isn't of our interest, however *can be useful in certain scenarios*. Such data usually describe optional information aboout the *payload* itself, or details of how it was delivered. Such data usually called *metadata*

> The information itself, that we wish to store or exchange colloquially called a *payload*. Some additional details, that might be useful, regarding that information, but not tied to it directly usually called *metadata*

Let's say, I am writing a document in Microsoft Word. The *payload* here being anything, I typed directly in this document. However, once I saved it, not only the document itself was saved, but also a bunch of additional *metadata*. It can include date of the document creation, author[9] of the document, last save date, last print date, etc. *The same logic is applicable for virtually any file you've ever created*[10].

[9] Usually currently active user on OS level

[10] Sometimes, ability to control metadata becomes crucial to save sensitive and personal information. Some professions can put you in physical danger, if you're not cautious enough

2.2 You can't manage what you can't measure

Once we have gotten ourselves these neat definitions about data and information, we will not benefit from them until we resolve one fundamental issue: *How do we measure information?* How can we possibly find an adequate solution to count something so abstract in nature.

Well, first of all this is where this fine distinction between *data* and *information* comes in play. You see, *defining context* to transform our data into information provided us with one *fundamental advantage*. To demonstrate this, let's consider an example: I have a drawer, where a bunch of my T-Shirts is stored. Let's say, there is no order whatsoever and once I open the drawer, any of my shirts can be on top[11]. Let's say that I for a fact know that there is no more than 12 T-Shirts of different colors totally in my drawer. I open a drawer and see a green T-Shirt on top. *How much information did I receive*?

To answer this question, let's investigate how any data is measured, for starters. Well, universally in information theory there is a number of data measurement nomenclatures. The most ubiquitous and universal one being *bits*. It got it's name from a 'binary digit'. The binary part represents a tiny set of values it can be equal to, consisting only of {0, 1}.

Bit is a unit of information. It can be equal to either '1' or '0'.

Bits are the most basic and fundamental part of any computer-related information operation. *Everything* on your computer is stored in bits. Yes, *everything*. Everything you type in your Word documents, PowerPoint presentations, every audio track you've ever played, every site you've ever visited, every file you've created, *everything*. This simple fact might leave us with a bit of confusion: *How on Earth can we measure apparently everything with such basic unit, only accepting '0' and '1' as a value?*. The answer is – well, pretty easily. Once we have something, that cannot possibly be presented as value in a set of {0, 1}, *we just increase the number of bits to store that additional information.*

Consider 1 bit. Total number of values it can represent is limited to 2, by definition. It's either '1' or '0'. Well, if we increase the total numbers of bits and consider we have 2 bits. Now we have 2 possible 'cells', each of which can 'hold' 2 possible variants. Those values being: '00', '01', '10', '11'. Now, if something is still bigger than one of four possible variants, we can add bits until it will finally be enough. With a little bit of discrete math we can say for sure, that total possible variants a sequence of bits can hold equals to 2^x, where 2 is a number of possible options a bit can 'hold' ('0', '1'), and x is the number of bits we are ready to *allocate*.

Number of bits	Maximum options	Options
1	2	{0, 1}
2	4	{00, 01, 10, 11}

[11] I would call that scenario uncanningly realistic

3	8	{000, 001, 010, 011, 100, 101, 110, 111}
...	...	...
x	2^x	{ x times 0, ..., x times 1}

To count something in bits is pretty similar as to how we would count something normally, but with one nuance. In a modern world, humans mostly calculating everything in a $base_{10}$. This simply means, that we have *10 digits* that construct *all of our numbers*[12]. Those digits being: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}. So, our *digits* can represent one of at-most 10 options. But how can we describe something more than 9? Well... just start over and add additional digit! This is the same principle we follow, when we are counting in a $base_2$, working with bits. We can use pretty much any base we want, but along with $base_2$, $base_8$ and $base_{16}$ can be seen used widely[13]. So we can pretty much convert number in any base to any other base, always keeping consistency:

$base_{10}$	$base_2$	$base_8$	$base_{16}$
0_{10}	0_2	0_8	0_{16}
1_{10}	1_2	1_8	1_{16}
2_{10}	10_2	2_8	2_{16}
3_{10}	11_2	3_8	3_{16}
4_{10}	100_2	4_8	4_{16}
5_{10}	101_2	5_8	5_{16}
6_{10}	110_2	6_8	6_{16}
7_{10}	111_2	7_8	7_{16}
8_{10}	1000_2	10_8	8_{16}
9_{10}	1001_2	11_8	9_{16}
10_{10}	1010_2	12_8	A_{16}
11_{10}	1011_2	13_8	B_{16}
12_{10}	1100_2	14_8	C_{16}
13_{10}	1101_2	15_8	D_{16}
14_{10}	1110_2	16_8	E_{16}
15_{10}	1111_2	17_8	F_{16}
16_{10}	10000_2	20_8	10_{16}
...	...	...	...

[12]There are examples of cultures, that used different number as their base. However $base_{10}$ is the ubiquitous one nowadays

[13]Writing everything in $base_2$ can be slightly inconvenient, when exchanging information with other programmers. Those bases gives us ability to 'shorten' binary code. Since bases are 2^3 and 2^4 respectively, we can 'group' together bits in a group of 3 (in the case of $base_8$) or 4 (in the case of $base_{16}$), making it easier for a human reading. Computer still operates with them in 'raw' $base_2$ format

This concept was wonderfully explained in Charles Petzold's book *Code: The Hidden Language of Computer Hardware and Software*. He explains different number bases with a following concept: we, humans, use $base_{10}$ mostly because it's very conveniently translates to *number of our fingers and toes*. So, a $base_2$ would be only natural, if math was invented by dolphins! Since they have only two flippers from each side it would be super convenient for them to calculate.

One might wonder: But why choose $base_2$ number system for computers in the first place? Well, the answer is, like many things, a result of overcoming real-world limitations. Any computer utilises some hardware, tangible, physical metal parts. Since computers nowadays are mostly electronic devices it made sense to create system detecting presence or absence of some electric signal. Presence of said signal would be connoted as '1' and absence as '0'.

Now, since we dwelled enough into subject, it's time to answer to the initial question of the subsection: *How much data did I receive, seeing a green T-Shirt on the top, once I opened my drawer*. We can even add some precision to this question, asking not *How much data ...*, but *How many bits of information ...*. Well, the fundamental *advantage* of data being in context, that I claimed in the beginning of the section is that now we know *total number of options*. Since we know, that I only have 12 T-Shirts (and we don't really care about any other clothing in the drawer), there 12 possible options this could've ended. To conclude the total number of bits this information will occupy, I can use something called *Hartley function* to find out. It can be represented[14] as: $2^x \geq N$ with N — representing our total outcomes number, 2 — representing possible options for one bit and x — *number of bits we need to store it*. Mind '$\geq$' here — since bit is indivisible measurement, we cannot just take 1.5 bits, for example. *All cases ending up with a non-integer number need to be rounded up.*

Since we know our total number of possible outcomes — 12, we can calculate total numbers of bits we need to store that information, being the nearest power of 2 exceeding possible outcomes number.

> $2^4 \geq 12 \implies$ we need *4 bits* to store information about *any* T-Shirt being on top.

Please, mind an important detail: we need 4 bits *regardless* of which exactly T-Shirt will end up on top. To represent which T-Shirt it was exactly in each case of me, opening my drawer (provided, that total number of T-Shirts is constant), we have to come up with a *code*.

> Code — a system of signals or symbols for communication[15].

We must assign to every one of my T-Shirts a special *system of symbols*. Since information is measured in bits, it's only natural for our 'symbols' to be simple bits sequences. Number of bits in these sequences will be equal to 4, since $2^4 \geq 12$
Let's come up with a system of codes for T-Shirts:

[14]This form is somewhat simplified for the sake of clarity. Full Hartley Formula can be denoted as: $H_0(A) := \log_b |A|$

[15]https://www.merriam-webster.com/dictionary/code

T-Shirt	Code
Red	0000
Brown	0001
Green	0010
Yellow	0011
Pink	0100
Gray	0101
Purple	0110
Blue	0111
Black	1000
White	1001
Orange	1010
Beige	1011

So, upon opening my drawer and seeing my green T-Shirt on top I received 4 bits of information. If my drawer (or the T-Shirt itself) could send me binary information, they would send me '0010' representing it's color in this particular context.

Thus, we created an *encoding* for our T-Shirts.

2.3 Size matters

2.4 Countess, weaving patterns and count machines

2.5 But can it speak?