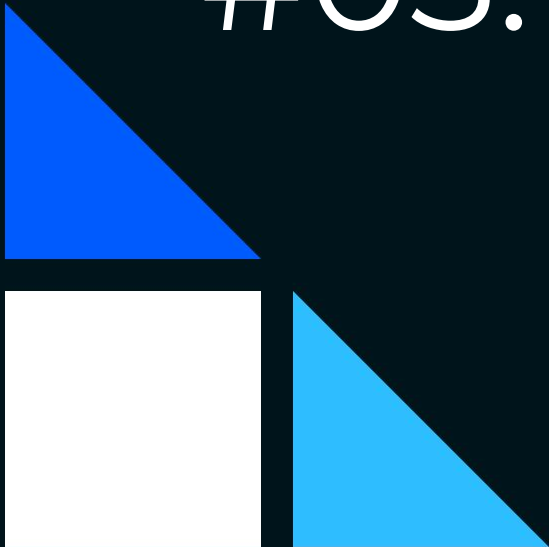


Blockchain In Rust

#03: Chains & Checks



geeklaunch

not a geek? start today!



Before we start

Download and install Rust if you want to code along: <https://www.rust-lang.org/>

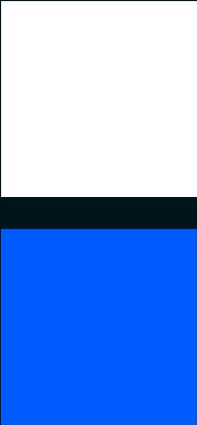
If you're on Windows, you'll also need Microsoft Visual C++ Build Tools 2017:
<https://visualstudio.microsoft.com/downloads/#build-tools-for-visual-studio-2017>.
Alternatively, it's *very easy* to install Rust on the Windows Subsystem for Linux (WSL).

If you're on Linux, you'll need some form of GCC, which probably came preinstalled on your system.

Optionally, you may also want to install Git: <https://git-scm.com/>

The code written in this series can be found at: <https://github.com/GeekLaunch/blockchain-rust>





Actually really finally making a blockchain for real (sorry it took so long)

You've watched over an hour of video—here's where we actually make our blocks into a chain.

Surprise!

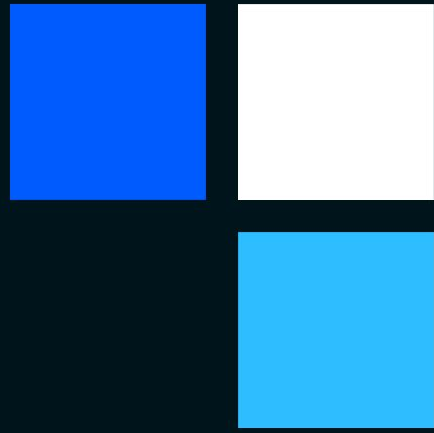
It's just a block array.

Well, vector.

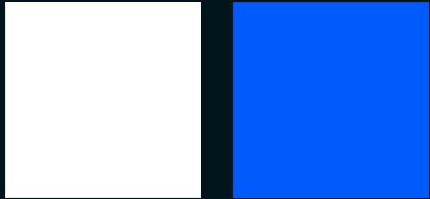
Well, struct containing a vector.

But it gets better...!





Implement: Blockchain Struct





Review: Mining

A block having been “mined” means that an amount of effort has been put into discovering a nonce “key” that “unlocks” the block’s hash-based “puzzle.”

Mining has the property that it is a hard problem to solve* while its solution is easy to check and verify.

* It has a customizable difficulty that should adapt to the amount of effort being put forth by the miners on the network to maintain the average time it takes to mine a block.

Bitcoin adjusts its difficulty every 2,016 blocks such that the next 2,016 blocks *should* take two weeks to mine.

https://en.bitcoin.it/wiki/Difficulty#What_network_hash_rate_results_in_a_given_difficulty.3F





Block Verification

Given the implementation we have so far, we can also implement a few rudimentary block verification tests. These steps would be executed whenever we receive a new block from a peer.

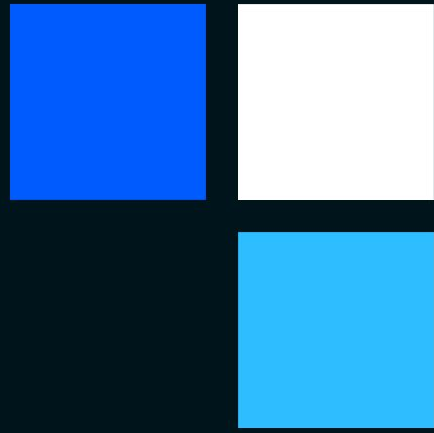
Each supposed valid block has a nonce attached to it that we *assume* took an approximately certain amount of effort to generate. This “approximately certain amount of effort” is described by the difficulty value. (Remember from the mining video?)

We will verify four things now:

1. Actual index == stored index value (note that Bitcoin blocks don't store their index)
2. Block's hash fits stored difficulty value (we'll just trust the difficulty for now) (*insecure*)
3. Time is always increasing (IRL network latency/sync demands leniency here)
4. Actual previous block's hash == stored prev_block_hash value (except for genesis block)

The Bitcoin protocol describes [these 19 verification steps for blocks](#).





Implement: Blockchain Verification

