AADHITHYA SIVAKUMAR

TIFAC-CORE IN CYBERSECURITY

AMRITA VISHWA VIDYAPEETHAM

SOLIDITY

OBJECT ORIENTED PROGRAMMING LANGUAGE

19CSE100 – PROBLEM SOLVING AND ALGORITHMIC THINKING PROGRAMMING LANGUAGE SURVEY ASSIGNMENT

ABOUT THE LANGUAGE

The Solidity programming language is an open-source, community project governed by a core team.

Solidity is an High-Level object-oriented programming language for implementing smart contracts on various blockchain platforms, most notably, Ethereum.

It is a statically typed programming language.

The Solidity proposal also includes "Natural Language Specification", a documentation system for specifying user-centric descriptions of the ramifications of method-calls.

HISTORY

Solidity was proposed in August 2014 by *Gavin Wood*.
The language was later developed by the Ethereum project's Solidity team, led by *Christian Reitwiessner*.

The team was led by Christian Reitwiessner, and Alex Beregszaszi, and several former Ethereum core contributors formed the team which was instrumental in developing solidity.

TIMELINE OF SOLIDITY DEVELOPMENT

- August 2014: The Solidity language was proposed by Gavin Wood.
- October 2014: Solidity was adopted as a language by Monax, a rival platform.
- August 2015: Solidity was officially released.

What is so special about it?

- Solidity is the code behind Ethereum – one of the largest blockchain platforms in the world (the other being Bitcoin).
- Bitcoin's currency (Bitcoin) and Ethereum's currency (Ether) are two of the most important and broadly traded cryptocurrencies available.
- The Bitcoin network was created as a peer-to-peer currency exchange. Bitcoin, as a token, was always intended to be a store of value – an asset. On the other hand, Ethereum was created as a way for people to move anything of value, efficiently, not just cryptocurrency.
- The token, Ether, was created as a way to pay for those dealings on the platform.
- Since it was too complex to deal with anything other than cryptocurrencies on the Bitcoin platform, the team at Ethereum created a new system. It required a computing language with far more flexibility than Bitcoin's (largely written in C++); so the team wrote the language Solidity.

PROGRAMMING PARADIGM

SOLIDITY is influenced by C++, Python and JavaScript.

SOLIDITY belongs to the object-oriented programming group under the imperative paradigm categorization of programming languages.

Solidity is a statically-typed curly-braces programming language designed for developing smart contracts that run on the Ethereum Virtual Machine.

WHY SOLIDITY SHOULD BE USED?

Bitcoin initiated the boom of blockchain technology. It was a quick and simple way to transact in a decentralized manner since there is no involvement from banks. Additionally, there is a record of every transaction in the unalterable public ledger so people have confidence in it. However, Bitcoin has its limits. If blockchain technology had the right programming, it could be used for much more than just peer-to-peer financial trades. So, Ethereum came into existence to overcome this challenge with the new programming language solidity.

It is a high-level programming language that is quite easy for developers to write and understand. Additionally, solidity is not just a programming language, it is also a tool to create machine-level code and compile it within Ethereum Virtual Machine (EVM). Essentially, developers use it for writing smart contracts on blockchain platforms to create business logic.

Solidity programming

1. The programming language gives developers the skill to create their own decentralised apps (dApps). Just as apps in the Apple app store are built to run on iOS, dApps on Ethereum are built to run on Solidity.

2. Within dApps, there are pieces of code recognised as smart contracts. These assist people to swap money, shares, property and nearly anything of value when certain conditions are met. That eliminates the need for an expensive third party like a notary.

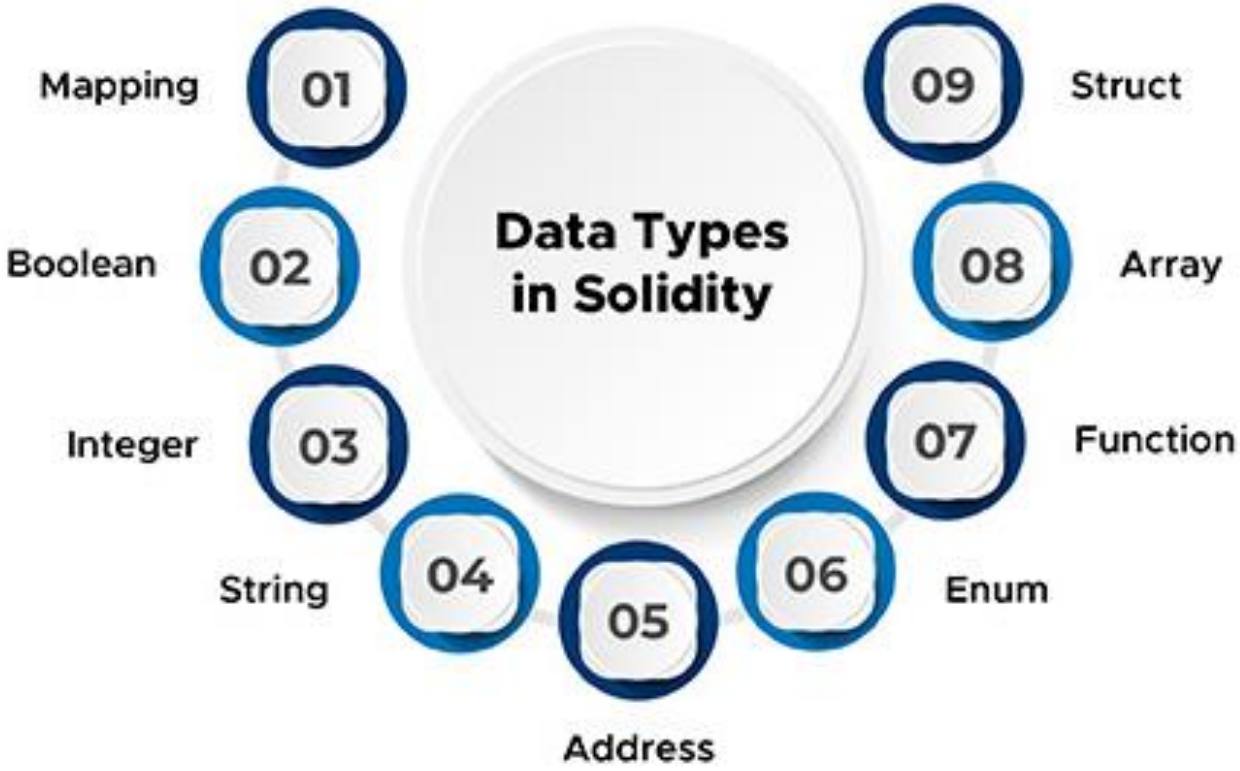
WHAT IS SMART-CONTRACT?

A smart contract is a computer program that runs on a blockchain platform to execute a task after fulfilling a pre-requisite condition. They work on the “if/when…then…” assertion. So, once a smart contract gets information from the ledger about a transaction, it performs the associated task automatically.

Smart contracts are self-executing contracts that work on the blockchain. Once the blockchain ledger confirms that necessary transactions have taken place, smart contracts execute their predetermined conditions. You can use them for financial transactions, supply chain management, and identity verification amongst other things.

Solidity smart contracts are a particularly safe way to transfer anything of value because they are created on a blockchain and no entity can change or destroy them.

DATA TYPES SUPPORTED BY SOLIDITY



A Simple Smart Contract - Storage Example

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.4.16 <0.9.0;

contract SimpleStorage {
    uint storedData;

    function set(uint x) public {
        storedData = x;
    }

    function get() public view returns (uint) {
        return storedData;
    }
}
```

What does the contract do?

This contract is a collection of code (its functions) and data (its state) that resides at a specific address on the Ethereum blockchain.

The line *uint storedData;* declares a state variable called storedData of type uint (unsigned integer of 256 bits).

In this example, the contract defines the functions *set* and *get* that can be used to modify or retrieve the value of the variable.

The first line tells you that the source code is licensed under the GPL version 3.0.
The next line specifies that the source code is written for Solidity version 0.4.16, or a newer version of the language up to, but not including version 0.9.0. This is to ensure that the contract is not compilable with a new (breaking) compiler version, where it could behave differently.
Pragmas are common instructions for compilers about how to treat the source code

Drawback

Solidity smart contracts require flawless and error-free programming.

You cannot go back and undo transactions. Changing smart contract processes is almost impossible, any error in the code can be time-consuming and expensive to correct.

The only method to address flaws is to update the Ethereum code, which is a complex task.

An ongoing issue with Solidity is its vulnerability to attack.

Smart contracts are vulnerable to phishing and other types of attacks as well.

Applications

Few examples of where solidity can be used are as given below

Voting: In the real world, the voting process is vulnerable to many forms of fraud. To solve a few of theproblems, we could make use of contracts for voting. Solidity can be used to plan the code, and with proper implementation, the process of voting will be smooth, transparent and automatic.

Crowdfunding: For crowdfunding, smart contracts can work out far better than non-trusted federal systems. These smart contracts can be developed using Solidity.

Blind auctions: Implementation of blind auctions using Solidity is quite simple on Ethereum. An open sale can be created, in which every person is aware of each other's bid. Following this, a blind auction can be designed whereby it will not be possible for anyone to see others' bids until the bidding process ends.

REFERENCES

https://www.opensourceforu.com/2019/08/an-introduction-to-solidity-the-language-that-runs-ethereum/

https://docs.soliditylang.org/en/v0.8.17/

https://soliditylang.org/about/#:~:text=The%20Solidity%20programming%20language%20is,governed%20by%20a%20core%20team.

https://blockchain.oodles.io/blog/solidity-programming-language-smart-contracts-development/

https://www.bairesdev.com/blog/why-use-solidity-for-blockchain-development/