**Blockchain Study Notes Day 11:**

**Module 2 - Solidity Basics**  
**Chapter 7 - Conditions in Solidity**

**Introduction to Conditions in Solidity**

Conditions in Solidity are essential for controlling the flow of execution in smart contracts. They help make decisions by evaluating expressions and executing code based on the results.

**1. Conditional Statements in Solidity**

**1.1. if Statement**

* Executes a block of code if a specified condition is true.
* **Syntax**:

if (condition) {

// Code to execute if condition is true

}

* **Example**:

function checkValue(uint \_value) public pure returns (string memory) {

if (\_value > 100) {

return "Value is greater than 100";

}

return "Value is less than or equal to 100";

}

**1.2. if...else Statement**

* Executes one block of code if the condition is true and another if it is false.
* **Syntax**:

if (condition) {

// Code if condition is true

} else {

// Code if condition is false

}

* **Example**:

function checkEvenOdd(uint \_value) public pure returns (string memory) {

if (\_value % 2 == 0) {

return "Even";

} else {

return "Odd";

}

}

**1.3. if...else if...else Statement**

* Allows checking multiple conditions sequentially.
* **Syntax**:

if (condition1) {

// Code if condition1 is true

} else if (condition2) {

// Code if condition2 is true

} else {

// Code if none of the conditions are true

}

* **Example**:

function grade(uint score) public pure returns (string memory) {

if (score >= 90) {

return "A";

} else if (score >= 75) {

return "B";

} else if (score >= 50) {

return "C";

} else {

return "F";

}

}

**2. Ternary Operator**

* A shorthand for if...else.
* **Syntax**:

condition ? trueExpression : falseExpression;

* **Example**:

function checkSmallNumber(uint \_value) public pure returns (string memory) {

return \_value < 10 ? "Small number" : "Not a small number";

}

**3. Example Program Using Conditions (Using Munawar)**

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract MunawarConditions {

// Function to check if Munawar is eligible to vote

function checkVotingEligibility(uint age) public pure returns (string memory) {

if (age >= 18) {

return "Munawar is eligible to vote";

} else {

return "Munawar is not eligible to vote";

}

}

// Function to categorize Munawar’s experience level

function experienceLevel(uint years) public pure returns (string memory) {

if (years >= 10) {

return "Expert";

} else if (years >= 5) {

return "Intermediate";

} else {

return "Beginner";

}

}

// Function using a ternary operator

function isMunawarHappy(bool hasCoffee) public pure returns (string memory) {

return hasCoffee ? "Munawar is happy" : "Munawar needs coffee";

}

}

**4. Best Practices for Using Conditions**

* **Simplify Logic**: Use else if and ternary operators to reduce redundant if statements.
* **Avoid Deep Nesting**: For better readability, avoid deeply nested if...else blocks.
* **Gas Efficiency**: Minimize complex conditions to reduce gas costs.

**Home Task**

1. **Enhance the Example Program**:
   * Add a function to check if a given year is a leap year using conditions.
2. **Write a New Contract**:
   * Implement a contract with a function to determine the largest of three numbers using if...else if...else.
3. **Experiment with Ternary Operators**:
   * Rewrite a simple if...else function using a ternary operator.

**Conclusion**

Conditions in Solidity are critical for decision-making within smart contracts. By understanding and applying if, else, and ternary operators effectively, developers can build dynamic and responsive blockchain applications.

Day 11 Notes

***Prepared by Munawar Johar***