```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.30;

import "./IERC20.sol";

abstract contract ERC20 is IERC20 {
string public name;
string public symbol;
uint8 public decimals;

uint256 public totalSupply;

mapping(address => uint256) public balanceOf;
//function allowance(address _owner, address _spender) external view returns
(uint256 remain);
mapping(address => mapping(address => uint256)) public allowance;

//https://youtu.be/w9778DlUsAQ?list=PL_r-IuCAlAeeI21VCmwdOWBZhPPwKs2Qz&t=1633
///https://youtu.be/dVxu0alH2ys?list=PL_r-IuCAlAeeI21VCmwdOWBZhPPwKs2Qz&t=20
///@dev quiere hacer un constructo para que genere token diferentes.

constructor(string memory _name, string memory _symbol) {
    name = _name;
    symbol = _symbol;
    decimals = 18;
    ///@dev aca el totalSupply (que son 100 token!! con 18 decimales. se le
asigna al creador , osea al owner que es el  msg.sender
    totalSupply = 100 * 10 ** 18;
    balanceOf[msg.sender] = totalSupply;
}

// Etiquetas NatSpec importantes

//     @title: Título descriptivo del contrato o función.
//     @author: Nombre del autor del contrato.
//     @notice: Describe la función al usuario final.
//     @dev: Proporciona detalles adicionales para desarrolladores.
//     @param: Documenta un parámetro específico de una función.
//     @return: Describe el valor de retorno de una función.

// function transfer(
//     address _to,
//     uint256 _value
// ) external returns (bool success) {
//     ///@dev - Usas requires cuando se dijo que no los usen y usen solo
custom errors. (fundamental)
//     // require(_to!=address(0), "no puedes enviar a la direccion 0");
//     // require(balanceOf[msg.sender]>=_value, "no tienes token para
```

```
transferir");
//     // balanceOf[msg.sender]-=_value; /// @notice: aca se pasa , revivierte
por eso hay que hacer la solicion mas abajo!!
//     // balanceOf[_to]+=_value;  ///@notice:
//     // return true;

//     ///@notice: se eliminan los require y se ponen los uncheked
//     // require(_to!=address(0), "no puedes enviar a la direccion 0");
//     // require(balanceOf[msg.sender]>=_value, "no tienes token para
transferir");

//     if (balanceOf[msg.sender] >= _value) revert();

//     /// @notice: aca se pasa , revivierte por eso hay que hacer la solicion
mas abajo!!
//     unchecked {
//         balanceOf[msg.sender] -= _value;
//     }
//     balanceOf[_to] += _value;
//     return true;
// }


/// @dev la de arriba ahora le vamos aplicar una funcion interna para que no
pueda ser modificada afuera ::
function _transfer(
    address _from,
    address _to,
    uint256 _value
) internal  virtual  returns (bool success) {

     // ❌ ESTO ESTÁ MAL - revierte cuando TIENE suficiente balance
      // if (balanceOf[_from] >= _value) revert();

    // ✅ ESTO ES CORRECTO - revierte cuando NO TIENE suficiente balance
    if (balanceOf[_from] < _value) revert();
    /// @notice: aca se pasa , revivierte por eso hay que hacer la solicion mas
abajo!!
    unchecked {
        balanceOf[_from] -= _value;
    }
    balanceOf[_to] += _value;
    emit  Transfer( _from,  _to,  _value);
    return true;
}
```

///@dev funcion transfer mejorada con funcion internal
function transfer(
address _to,

uint256 _value
) external returns (bool success) {
return _transfer(msg.sender,_to,_value);
}

///@dev tambien la optimizamos ==
// function transferFrom(
// address _from,
// address _to,
// uint256 _value
//) external returns (bool success) {

```
//      if (balanceOf[_from] >= _value) revert();
//     /// @notice: aca se pasa , revivierte por eso hay que hacer la solicion
mas abajo!!
//     unchecked {
//         balanceOf[_from] -= _value;
//     }
//     balanceOf[_to] += _value;
//     return true;
// }
```

/// @dev ahora tenes que verificar que la persona que transfiere tenga el allowance habilitado
function transferFrom(
address _from,
address _to,
uint256 _value
) external returns (bool success) {
if(_value==0)revert ZeroValue();
if(allowance[_from][msg.sender]<_value) revert InvalidadAllowances();//owner y el spender(es el que ejecuta esa funcion .
unchecked {
allowance[_from][msg.sender]-=_value;
}
return _transfer(_from,_to,_value);
}

```
///@dev esta funcion es para aprobar a un spender que pueda ejecutar
transferFrom (en realidad no lo aprueba
//setea lo que tiene permitido gastar
function approve(address _spender, uint256 _value) external returns (bool
success){
allowance[msg.sender][_spender]=_value;
emit Approval(msg.sender, _spender,  _value);
```

```
    return true;

    }
```

}