Here's a basic Solidity smart contract for recording carbon credits:

This contract:

- **Tracks carbon credits with project details**
- **Allows issuance of new credits**
- **Enables transfer of credits between addresses**
- **Provides read functions to check credit details and ownership**

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

contract CarbonCreditRegistry {
    struct CarbonCredit {
        uint256 amount;
        string projectId;
        uint256 issuanceDate;
        address owner;
    }

    CarbonCredit[] private _carbonCredits;
    mapping(address => uint256[]) private _ownedCredits;

    event CreditIssued(uint256 indexed creditId, address indexed owner, uint256 amount, string projectId);
    event CreditTransferred(uint256 indexed creditId, address indexed from, address indexed to);

    function issueCredit(
        address owner,
        uint256 amount,
        string memory projectId
    ) external {
        uint256 creditId = _carbonCredits.length;
        _carbonCredits.push(CarbonCredit({
            amount: amount,
            projectId: projectId,
            issuanceDate: block.timestamp,
            owner: owner
        }));

        _ownedCredits[owner].push(creditId);
        emit CreditIssued(creditId, owner, amount, projectId);
    }

    function transferCredit(
        uint256 creditId,
        address newOwner
    ) external {
        require(creditId < _carbonCredits.length, "Invalid credit ID");
        CarbonCredit storage credit = _carbonCredits[creditId];
        require(msg.sender == credit.owner, "Not the owner");

        // Remove from current owner's list
        uint256[] storage ownerCredits = _ownedCredits[credit.owner];
```

```
        for (uint256 i = 0; i < ownerCredits.length; i++) {
            if (ownerCredits[i] == creditId) {
                ownerCredits[i] = ownerCredits[ownerCredits.length - 1];
                ownerCredits.pop();
                break;
            }
        }

        // Update owner and add to new owner's list
        credit.owner = newOwner;
        _ownedCredits[newOwner].push(creditId);

        emit CreditTransferred(creditId, msg.sender, newOwner);
    }

    function getCreditDetails(uint256 creditId) external view returns (CarbonCredit memory) {
        require(creditId < _carbonCredits.length, "Invalid credit ID");
        return _carbonCredits[creditId];
    }

    function getOwnedCredits(address owner) external view returns (uint256[] memory) {
        return _ownedCredits[owner];
    }
}
```