Here is a C++ solution that first writes the ghost legs ladder to a file and then reads it back into a vector<vector<string>>. This approach is useful for saving and loading game states or other grid-based data.

Writing the Ghost Legs to a File

To write the ghost legs data, you can use std::ofstream from the <fstream> library. This allows you to create and write to a text file. The easiest format is to represent each row of the ladder as a line in the file, with the individual column elements separated by spaces.
The following code defines a sample ghost legs ladder and writes it to a file named ghost_legs.txt.

C++

```
#include <fstream>
#include <iostream>
#include <vector>
#include <string>

// Function to write the ghost legs ladder to a file
void writeLadderToFile(const std::vector<std::vector<std::string>>& ladder, const std::string&
filename) {
  std::ofstream outFile(filename);
  if (!outFile) {
    std::cerr << "Error: Could not open the file for writing." << std::endl;
    return;
  }

  for (const auto& row : ladder) {
    for (const auto& element : row) {
      outFile << element << " ";
    }
    outFile << "\n";
  }

  outFile.close();
  std::cout << "Successfully wrote ghost legs to '" << filename << "'" << std::endl;
}
```

The writeLadderToFile function iterates through each row and each element, writing the content to the file followed by a space. It then writes a newline character at the end of each row. A basic error check ensures the file was opened successfully.

Reading the Ghost Legs from a File

Reading the data back requires std::ifstream and std::stringstream. We'll read the file line by line, and for each line, we'll parse the individual strings using a string stream.

C++

```
#include <fstream>
#include <iostream>
#include <vector>
#include <string>
#include <sstream>

// Function to read the ghost legs ladder from a file
std::vector<std::vector<std::string>> readLadderFromFile(const std::string& filename) {
    std::vector<std::vector<std::string>> ladder;
    std::ifstream inFile(filename);

    if (!inFile) {
        std::cerr << "Error: Could not open the file for reading." << std::endl;
        return ladder;
    }

    std::string line;
    while (std::getline(inFile, line)) {
        if (line.empty()) {
            continue; // Skip empty lines
        }
        std::stringstream ss(line);
        std::string element;
        std::vector<std::string> row;

        while (ss >> element) {
            row.push_back(element);
        }
        ladder.push_back(row);
    }
    inFile.close();
    std::cout << "Successfully read ghost legs from '" << filename << "'" << std::endl;
    return ladder;
}
```

The readLadderFromFile function reads each line into a string, then uses a std::stringstream to extract individual elements separated by spaces. It populates a vector<vector<string>> and returns it.

Putting It All Together

You can combine these two functions in a main function to demonstrate the entire process. First, the ladder is written to a file, and then it is read back and printed to the console to verify that the data was restored correctly.

C++

```cpp
#include <fstream>
#include <iostream>
#include <vector>
#include <string>
#include <sstream>

// Function to write the ghost legs ladder to a file
void writeLadderToFile(const std::vector<std::vector<std::string>>& ladder, const std::string&
filename) {
    std::ofstream outFile(filename);
    if (!outFile) {
        std::cerr << "Error: Could not open the file for writing." << std::endl;
        return;
    }

    for (const auto& row : ladder) {
        for (const auto& element : row) {
            outFile << element << " ";
        }
        outFile << "\n";
    }
    outFile.close();
    std::cout << "Successfully wrote ghost legs to '" << filename << "'" << std::endl;
}

// Function to read the ghost legs ladder from a file
std::vector<std::vector<std::string>> readLadderFromFile(const std::string& filename) {
    std::vector<std::vector<std::string>> ladder;
    std::ifstream inFile(filename);

    if (!inFile) {
        std::cerr << "Error: Could not open the file for reading." << std::endl;
        return ladder;
    }

    std::string line;
    while (std::getline(inFile, line)) {
        if (line.empty()) {
            continue; // Skip empty lines
        }
        std::stringstream ss(line);
        std::string element;
        std::vector<std::string> row;

        while (ss >> element) {
            row.push_back(element);
        }
        ladder.push_back(row);
    }

    inFile.close();
    std::cout << "Successfully read ghost legs from '" << filename << "'" << std::endl;
    return ladder;
}

int main() {
    // 1. Create a sample ghost legs ladder
    std::vector<std::vector<std::string>> ghostLegsLadder = {
        {"|", "", "|", "", "|"},
        {"|", "-", "|", "", "|"},
        {"|", "", "|", "-", "|"},
        {"|", "-", "|", "", "|"},
        {"|", "", "|", "", "|"}
    };
    const std::string filename = "ghost_legs.txt";

    // 2. Write the ladder to a file
    writeLadderToFile(ghostLegsLadder, filename);

    // 3. Read the ladder back from the file
    std::vector<std::vector<std::string>> loadedLadder = readLadderFromFile(filename);

    // 4. Print the loaded ladder to verify
    if (!loadedLadder.empty()) {
        std::cout << "\nVerifying the loaded ladder from file:" << std::endl;
        for (const auto& row : loadedLadder) {
            for (const auto& element : row) {
                // A ternary operator is used to handle empty strings for better readability
                std::cout << (element.empty() ? """" : element) << " ";
            }
            std::cout << std::endl;
        }
    }

    return 0;
}
```

This code snippet provides a complete and reusable solution for saving a 2D vector of strings to a file and loading it back, which is a common task in C++ development.