📘 Deserializing C++ Classes from a CSV File

🧾 What is Deserialization?

Deserialization is the process of converting data from a storage format (like a CSV file) back into an in-memory object. In C++, this typically involves reading the file line-by-line, parsing the data, and populating class instances.

📄 CSV Format Overview

A **CSV (Comma-Separated Values)** file stores tabular data in plain text. Each line represents a record, and each field is separated by a comma:

```
id,name,age
1,Alice,22
2,Bob,25
```

🧱 Deserializing into a Class

1. Define the Class

```cpp
#include <string>

class Person {
    int id;
    std::string name;
    int age;
public:

    Person(int i, const std::string& n, int a) : id(i), name(n), age(a) {}
    Person(const std::string& csvData, char delimiter)
    {
        DeserializeCSV(csvData, delimiter);
    }

    void DeserializeCSV(const std::string& csvData, char delimiter);
};

void Person::DeserializeCSV(const std::string& csvData, char delimiter)
{
    std::stringstream ss(csvData);
    std::string token;

    std::getline(ss, token, delimiter);
    id = std::stoi(token);

    std::getline(ss, name, delimiter);

    std::getline(ss, token, delimiter);
    age = std::stoi(token);
}
```

2. Read and Parse the CSV File

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

std::vector<Person> deserializeCSV(const std::string& filename) {
    std::vector<Person> people;
    std::ifstream file(filename);
    if(file.is_open())
    {
        std::string line;

        // Skip header
        std::getline(file, line);

        while (std::getline(file, line)) {
            Person nextPerson(line, ',');
            people.push_back(nextPerson);
        }
    }
    else
        std::cout << fileName << " could not be opened.\n";

    file.close();

    return people;
}
```

3. Use the Deserialized Data

```cpp
int main() {
    std::vector<Person> people = deserializeCSV("people.csv");

    for (const auto& person : people) {
        std::cout << "ID: " << person.id
                  << ", Name: " << person.name
                  << ", Age: " << person.age << std::endl;
    }

    return 0;
}
```

🛠️ Notes and Best Practices

- **Error Handling**: Always check if the file opened successfully.

🎯 Quiz!

Here's a short quiz on the topic: quiz

Footer Separator

🔭 Markdown Viewer

How to view the markdown files in a browser...

- Markdown Viewer

🧠 Lecture Practices

Here are the lecture Practices...

- Day 10
- Day 11

🔍 Lecture Quizzes

Here are the lecture quizzes...

- Day 10
- Day 11

Weekly Topics

Here are the topics for the week...

- CSV
- Writing CSV
- Reading CSV
- Serializing
- Deserializing