Course Project Documentation

Project Topic: JSON/XML Simple Library (JXSL)

- ## Overview

This project implements a library for managing JSON and XML data in both C and C++ environments. The library provides functionality for adding, editing, deleting, and reading data from JSON and XML.

Additionally, the library supports optimized file operations and cross-verification of results between C and C.

- ## Project Directory Structure

```
JXSL_Project/
JXSL_C/                    # C Implementation
jxsl_lib.c                 # Main implementation file (C)
jxsl_lib.h                 # Header file (C)
jxsl_lib_test.c            # Test file (C)
JXSL_CPP/
                           # C++ Implementation
jxsl_lib_cpp.cpp   # Main implementation file (C++)
jxsl_lib_cpp.h             # Header file (C++)
jxsl_lib_cpp_test.cpp # Test file (C++)
jxsl_cross_test.cpp  # Cross-verification test file (C and C++)
README.md                  # Project README
CMakeLists.txt             # CMake configuration
```

- Project Structure

```
JXSL_Project/
├── JXSL_C/                # C Implementation
│   ├── jxsl_lib.c         # Main implementation file (C)
│   ├── jxsl_lib.h         # Header file (C)
│   ├── jxsl_lib_test.c    # Test file (C)
├── JXSL_CPP/              # C++ Implementation
│   ├── jxsl_lib_cpp.cpp   # Main implementation file (C++)
│   ├── jxsl_lib_cpp.h     # Header file (C++)
│   ├── jxsl_lib_cpp_test.cpp # Test file (C++)
│   ├── jxsl_cross_test.cpp  # Cross-verification test file (C and C++)
├── README.md              # Project README
├── CMakeLists.txt         # CMake configuration
```

C Files

• jxsl_lib.c: Core functions for working with JSON and XML in C.

• jxsl_lib.h: Header exposing the C functions.

• jxsl_lib_test.c: Contains unit tests for the C implementation.

C++ Files

• jxsl_lib_cpp.cpp: Core functions for working with JSON and XML in C++ as a class.

• jxsl_lib_cpp.h: Header exposing the C++ methods.

• jxsl_lib_cpp_test.cpp: Contains unit tests for the C++ implementation.

• jxsl_cross_test.cpp: Verifies consistency between the C and C++ implementations.

- Used Libraries

• Standard Libraries:

  - <iostream>: For input and output operations in C++.
  - <fstream>: For file handling in C++.
  - <cstdio>: For file handling in C.
  - <cstring>: For manipulating strings in C.
  - <vector>: For dynamic arrays in C++.
- Windows-specific APIs:
  - <windows.h>: For memory-mapped file operations.
- Class Descriptions

Class JXSL (C++)

Implements functionality for managing JSON and XML files as a class in C++.

Attributes:

- std::string filename: The file being managed.
- std::vector<std::pair<std::string, std::string>> data: Stores key-value pairs in memory.

Methods:

- bool add_data(const std::string& key, const std::string& value): Adds a key-value pair.
- bool edit_data(const std::string& key, const std::string& new_value): Edits the value for a key.
- bool delete_data(const std::string& key): Deletes a key-value pair.
- bool read_data(const std::string& key, std::string& value): Reads the value for a key.
- void display_data() const: Displays all stored data.
- void flush(): Writes in-memory changes to the file.

- # How to Build and Run

```
cmake -S . -B build
cmake --build build
```

Building with CMake

- Open a terminal in the project root directory.

Running Tests

- C Implementation Tests:

  ./build/jxsl_lib_test

- C++ Implementation Tests:

  ./build/JXSL_CPP

- Cross-Verification Tests:

  ./build/jxsl_cross_test

  Example usage (CPP):

  ```
  JXSL handler("test.json");
  handler.add_data("key1", "value1");
  handler.edit_data("key1", "new_value");
  handler.delete_data("key1");
  handler.flush();

  C:
  add_data_json("test.json", "key1", "value1");
  edit_data_json("test.json", "key1", "new_value");
  delete_data_json("test.json", "key1");
  ```

Author: Oleksandr Shchur