Writing and reading files

```
/*


    . Topics:
        . Writing to files
        . Reading from files
        . Text
        . Binary format
        . std::filesystem
        . std::ofstream
        . std::ifstream
        . stream read methods
        . stream write methods


*/
```

Writing and reading text files

```
//Write text format
export void write_file(const std::filesystem::path& file_path, const std::string& content) {

    //Open the file for writing and in append mode
    //Creating the stream object attempts to open the file
    std::ofstream file(file_path, std::ios::out | std::ios::app);
    if (!file.is_open()) {
        fmt::println("Failed to open file for writing: {}", file_path.filename().string());
        return;
    }

    //Write to the file using one of the write functions
    //Se comment above for a few other ways we could write to the file
    file << content;

    //It's good practice to close the file when done
    //The file is closed when the file object goes out of scope
    //The destructor of the file object closes the file
}
```

Writing and reading text files

```
//Read text format
export void read_file(const std::filesystem::path& file_path) {

    std::ifstream file(file_path, std::ios::in); // Open for reading
    if (!file.is_open()) {
        fmt::println("Failed to open file for reading: {}", file_path.filename().string());
        return;
    }

    /*
        . The std::getline function reads a line from the file and stores it in
            the string line. This proces is repeated until there are no more lines to read.
    */
    std::string line;
    while (std::getline(file, line)) {
        fmt::println("{}", line);
    }
}
```

Writing and reading text files

```cpp
//Read text format
export void read_file(const std::filesystem::path& file_path) {
    std::ifstream file(file_path, std::ios::in); // Open for reading
    if (!file.is_open()) {
        fmt::println("Failed to open file for reading: {}", file_path.filename().string());
        return;
    }

    /*
        . The std::getline function reads a line from the file and stores it in
            the string line. This proces is repeated until there are no more lines to read.
    */
    std::string line;
    while (std::getline(file, line)) {
        fmt::println("{}", line);
    }
}
```

Writing and reading text files

```
std::filesystem::path file_path = R"(D:\sample_file.txt)"; // Windows
//std::filesystem::path file_path = R"(/path/to/your/input_file.txt)"; // Linux

std::string content = "\nHello, world!\nThis is a test.";

// Write content to the file
text_files::write_file(file_path, content);

// Read content back from the file and print it
//fmt::println("Content read from the file:");
text_files::read_file(file_path);
```

Writing and reading binary files

```
// Write the data in binary
export void write_file(const std::filesystem::path& file_path, const std::vector<char>& content) {
    std::ofstream file(file_path, std::ios::out | std::ios::binary); // Binary mode
    if (!file.is_open()) {
        fmt::print("Failed to open file for writing: {}\n", file_path.filename().string());
        return;
    }

    // Write the entire content to the file as binary
    // Knowing the size of the binary data is important.
    file.write(content.data(), content.size());
}
```

Writing and reading binary files

```cpp
// Read the data in binary
// Function to read binary data from a file
export std::vector<char> read_file(const std::filesystem::path& file_path) {
    std::ifstream file(file_path, std::ios::in | std::ios::binary); // Binary mode
    if (!file.is_open()) {
        fmt::print("Failed to open file for reading: {}\n", file_path.filename().string());
        return {};
    }

    // Get the file size by seeking to the end
    //file.seekg(0, std::ios::end);
    //std::streamsize size = file.tellg();
    //file.seekg(0, std::ios::beg);

    //Get the size of the file using std::filesystem::file_size
    std::streamsize size = std::filesystem::file_size(file_path);

    // Allocate a buffer and read the file into it
    std::vector<char> buffer(size);
    if (file.read(buffer.data(), size)) {
        return buffer;
    }

    return {};
}
```

Writing and reading binary files

```cpp
// Read the data in binary
// Define file path
std::filesystem::path file_path = R"(D:\sample_file.bin)"; // Windows
//std::filesystem::path file_path = R"(/path/to/your/input_file.bin)"; // Linux

// Example content to write (binary data)
std::vector<char> content = {'H', 'e', 'l', 'l', 'o', ' ', 'B', 'i', 'n', 'a', 'r', 'y', '!'};

// Write binary data to file
binary_files::write_file(file_path, content);

// Read binary data from file
std::vector<char> read_content = binary_files::read_file(file_path);

// Display the read content as text
if (!read_content.empty()) {
    fmt::print("Read content: {}\n", std::string(read_content.begin(), read_content.end()));
} else {
    fmt::print("No content read from file.\n");
}
```

Writing and reading binary files

```
/*

        . Examples:
            . writing and reading csv
            . Writing and reading json
*/
```