C++: Streams and Strings

for Geoinformatics

Depicted Candela

September 29, 2024

Chapter 4: Streams (GIS Applications Focused on Stream Concepts, Including Skills from Chapter 3: Strings)

1. Basic String Manipulation and Output Formatting

Task: Write a program that reads the full name of a city (including spaces) using `getline` and outputs it in uppercase using the `toupper` function from the `<cctype>` library.
Application: Display the city name in a neatly centered format on a 50-character wide line, simulating a header for a map display.

2. Reading and Writing GIS Data Files

Task: Create a program that reads a list of place names from a file and writes only the places that start with a vowel to another file. Use functions from Chapter 3 (`startsWith`) and Chapter 4 (file streams).
Application: This could simulate filtering place names for a specific GIS application, like displaying only certain locations on a map.

3. Parsing CSV Data for GIS

Task: Implement a function that reads a line of comma-separated values (CSV format) containing coordinates and place names and splits them into individual fields. Use the `getline` function with a delimiter to perform the split.
Application: Write a program that reads multiple lines of CSV data from a file (representing coordinates and place names) and outputs the parsed data to the console in a formatted table, simulating loading and displaying data in a GIS system.

4. Advanced String Operations in Stream Contexts

Task: Develop a program that reads a paragraph of text from a file describing geographic features, counts the frequency of each word (ignoring case and punctuation), and writes

the results to an output file in descending order of frequency.
Application: This can be used to analyze and summarize the content of GIS metadata or descriptions associated with map layers.

5. Stream Manipulators for Number Formatting

Task: Write a program that reads a list of floating-point numbers from a file representing latitude and longitude values, formats them to six decimal places using stream manipulators, and outputs them aligned in columns with specific width.
Application: This simulates formatting geographical coordinates for a printed report or console output in a GIS application.

6. Implementing a GIS Log File Processor

Task: Create a program that reads a GIS log file line by line, identifies lines containing specific keywords (e.g., "ERROR", "WARNING"), and writes those lines to a separate summary file.
Application: Use string operations to parse and match patterns, and streams to handle file I/O, ensuring the program can handle different line endings and encodings gracefully, which is critical in maintaining GIS system logs.

7. Error Handling with Streams in GIS Data Processing

Task: Develop a program that attempts to read from a series of GIS data files (names provided by the user) and logs any errors encountered during file operations to a separate log file.
Application: The program should continue processing the next file even if an error occurs, demonstrating robust error handling with streams, which is essential when dealing with large sets of GIS data files.

8. Creating a Simple GIS Text Formatter

Task: Build a text formatting tool that reads a file containing geographical descriptions and formats its content to a fixed width, adding line breaks appropriately without breaking words. Implement functions from Chapter 3 to handle string splitting and joining.
Application: Add options to justify the text to the left, right, or center within the specified width using stream manipulators, simulating formatted text output in GIS reports.

9. Interactive GIS Console Application Using Streams and Strings

Task: Write a command-line tool that continuously accepts user input, processes commands like "count locations", "reverse coordinates", and "save to file", and displays results using formatted output.

Application: Utilize streams for interaction and file I/O, and string manipulation functions for processing commands, providing a simple interactive GIS tool for data analysis.

Advanced Stream Manipulator Usage in GIS Contexts

1. Hexadecimal and Octal Formatting for Coordinate Systems

Task: Write a program that reads a coordinate value (as an integer, for simplicity) and displays it in decimal, hexadecimal, and octal formats using `dec`, `hex`, and `oct` manipulators.
Application: This could be used in debugging low-level GIS systems or map projections that require such formats.

2. Scientific Notation and Precision Control in GIS Data

Task: Create a program that reads a floating-point number representing elevation data and displays it using both fixed and scientific notation with a precision of four decimal places.
Application: This simulates displaying geospatial data with appropriate precision and format for scientific analysis in GIS.

3. Whitespace and Alignment Control in GIS Reports

Task: Implement a program that reads a line of text (e.g., a description of a geographic feature) and outputs it with and without skipping whitespace using the `noskipws` manipulator.
Application: Use `setw`, `left`, `right`, and `internal` manipulators to demonstrate different text alignments in the output, similar to formatting descriptions in GIS metadata.

Stream States and Error Handling Mechanisms in GIS

1. Stream State Check and Error Recovery in GIS File Processing

Task: Write a program that attempts to read integers representing coordinates from a file until the end of the file is reached. Use stream state-checking methods like `eof()`, `fail()`, `bad()`, and `good()` to handle various errors, and implement recovery mechanisms for non-critical errors.
Application: Demonstrate clearing the error state using `clear()` and explain its significance in the context of GIS data processing.

2. Robust Input Handling for GIS Data

Task: Develop a program that prompts the user to enter valid coordinates (floating-point numbers). If the user enters invalid data (e.g., letters or symbols), the program should detect the error using `fail()`, display an error message, clear the stream, and

prompt for input again.
Application: This ensures that only valid coordinate data is entered, crucial for maintaining data integrity in GIS applications.

Stream Buffer Manipulation in GIS

1. Redirecting Output Streams for GIS Logs

Task: Create a program that redirects the standard output to a GIS log file using `rdbuf()` and `ofstream`. Then, demonstrate restoring the standard output back to the console.
Application: Extend this exercise to redirect the standard error (`cerr`) to a log file while keeping regular output on the console, useful for debugging GIS applications.

Advanced Formatted Input Operations for GIS

1. Extracting Formatted Geospatial Data

Task: Write a program that reads date values representing data collection dates from a formatted string (e.g., "2024-09-17") using `>>` with `setfill` and `setw` to ensure proper extraction and handling of each date component (year, month, day).
Application: Include validation steps to confirm the format is correct, simulating the handling of timestamped GIS data.

Working with stringstream in GIS

1. Parsing and Formatting Geospatial Data with `stringstream`

Task: Implement a program that reads a line of text containing comma-separated values (e.g., "latitude,longitude,altitude") and uses `stringstream` to parse each value into a vector of strings.
Application: Extend the program to convert numerical string inputs into integers and floating-point numbers, handling potential conversion errors gracefully, which is crucial when dealing with coordinate data in GIS.

Binary File I/O for GIS

1. Reading and Writing Binary GIS Data

Task: Write a program that opens a binary file and reads data structures (e.g., a struct with an integer for place ID and a double for coordinates) using the `read()` method. Then, modify the data and write it back to another binary file using the `write()` method.
Application: Include checks to handle end-of-file and any I/O errors during the read and write operations, simulating handling of binary GIS data files.

Custom Manipulators for GIS

1. Creating Custom Manipulators for GIS Coordinates

Task: Design a custom manipulator called `coord` that formats coordinates with a fixed precision and adds a degree symbol after the values (e.g., 12.345678 becomes "12.345678°").
Application: Test the custom manipulator by integrating it into a larger program that formats coordinate data for display in a GIS system.

Locale and Internationalization in GIS

1. Using Locales for International GIS Data Formatting

Task: Create a program that formats coordinates and dates according to different locales (e.g., US, Germany, Japan). Use the `locale` class to set the locale for input and output streams, demonstrating how the formatting changes.
Application: Specifically, show how different locales handle decimal points, thousands separators, and date formats in GIS applications.

2. International Date and Number Parsing for GIS Data

Task: Write a program that reads a date and a coordinate from the user in a locale-specific format (e.g., dates in DD/MM/YYYY for Europe). Use the `locale` class to correctly parse these inputs and display them in a standardized format.
Application: This ensures compatibility and proper formatting of GIS data in international contexts.