Project 1 Zip Code

Team 4: Minh Quan Tran, Mitchell Rogers, Carson Kariniemi, Abel Asfaw, Mahad Farah

Instructor: Andrew Anda

Due date: 09/23/2025

Design Document

1. Introduction

This project is designed to parse, store, and display U.S. postal code data from a CSV file. The data includes information such as the ZIP code, place name, state, county, latitude, and longitude.

The program provides two main functionalities:

* Displaying postal codes sorted by ZIP code (main1.cpp).
* Displaying postal codes sorted by State (main2.cpp).

The design uses object-oriented programming principles in C++, encapsulating postal code data within a PostalCodeItem class and managing collections of these items with a PostalList class.

1. Data structures
   1. PostalCodeItem

Defined in PostalCodeItem.h and implemented in PostalCodeItem.cpp, this class represents a single postal code entry with the following attributes:

* int zip — ZIP code.
* string place — Place name.
* string state — State abbreviation.
* string county — County name.
* double latitude — Latitude coordinate.
* double longitude — Longitude coordinate.
  1. PostalList

Defined in PostalList.h and implemented in PostalList.cpp, this class manages a collection of PostalCodeItem objects using vector with the following attributes:

* vector<PostalCodeItem> items — internal storage of postal code entries.
  1. CSV Reader Utility

Defined in readCSV.cpp, the function inputCSVtoList(PostalList &list, string filename) reads postal code data from a CSV file (us\_postal\_codes.csv) and populates a PostalList. It uses ifstream to parse each line, extracts fields separated by commas, and assigns them to a PostalCodeItem.

1. Functions
   1. PostalCodeItem

Key methods:

* Constructors: PostalCodeItem(), PostalCodeItem(int, string, string, string, double, double).
* Getters: getZip(), getPlace(), getState(), getCounty(), getLatitude(), getLongitude().
* Setters: setZip(), setPlace(), setState(), setCounty(), setLatitude(), setLongitude().
* Utility: printInfo() — displays formatted postal code information.
  1. PostalList

Key methods:

* addItem() — adds an item to the list.
* getItem() — retrieves an item by index.
* findByZip() — finds an item by ZIP code.
* size() — returns the number of items.
* printAll() — prints all items.
* printSortedByZip() — prints items sorted by ZIP.
* printSortedByState() — prints items sorted by state and then ZIP.
  1. CSV Reader

Key methods:

* inputCSVtoList(PostalList &list, string fileName) — reads CSV, parses fields, creates PostalCodeItem objects, and adds them to the PostalList.
  1. Main Programs

For main1.cpp:

* Loads data from CSV.
* Prints a formatted table of postal codes sorted by ZIP code using printSortedByZip().

For main2.cpp:

* Loads data from CSV.
* Prints a formatted table of postal codes sorted by State using printSortedByState().

1. The main program
   1. Create a PostalList instance.
   2. Load CSV data into the PostalList using inputCSVtoList().
   3. Print headers for the output table.
   4. Call the appropriate PostalList function to display sorted results:

* printSortedByZip() in main1.cpp.
* printSortedByState() in main2.cpp.
  1. End program execution.

1. Conclusion

The project provides a structured, object-oriented approach to handling postal code data.

* Encapsulation: The PostalCodeItem class encapsulates postal data.
* Data management: The PostalList class manages and organizes collections.
* Extensibility: Additional sorting or searching methods (e.g., by county, latitude) could be easily added.
* Separation of concerns: Data parsing is handled independently in readCSV.cpp.

This modular design ensures maintainability and scalability for future enhancements, such as integrating user queries, filtering options, or graphical output.