P2 Design Document

Preliminary Documentation for Buffer.h and Buffer.cpp

Overview:

The `Buffer` class is responsible for **reading, parsing, and storing Zip Code data** from files. It also supports reading from **length-indicated binary files** and grouping records by **state** for efficient access.

Purpose:

- **Store Zip Code Data:**
- Holds each Zip Code's **ID, city, state ID**, and **geographic coordinates**.
- **Process CSV Files:**

Reads Zip Code records from `us_postal_codes.csv`.

- **Read Length-Indicated Files:**

Supports binary format records using **length indicators**.

- **Organize by State:**

Allows records to be grouped by **state ID**.

Key Components:

- **`ZipCodeRecord` Struct:**

  Fields: `zip_code`, `city`, `state_id`, `latitude`, `longitude`
- **Buffer Class:**

  bool read_csv();

Reads the us_postal_codes.csv file and stores Zip Code records.

  std::map<std::string, std::vector<ZipCodeRecord>> get_state_zip_codes() const;

Groups the records by **state** and returns a map.

  ZipCodeRecord parse_csv_line(const std::string& line) const;

Parses individual lines of the CSV into ZipCodeRecord objects.

  bool readLengthIndicatedRecord(std::ifstream& fileStream, ZipCodeRecord& record);

Reads a length-indicated binary record from the input file.

Preliminary Documentation for CSVProcessing.h and CSVProcessing.cpp

Overview:

The **CSVProcessing class** handles the **sorting, filtering, and exporting of Zip Code data** from the buffer. It finds the **northernmost, southernmost, easternmost, and westernmost points** for each state and outputs the sorted data to CSV files.

Purpose:

- **Sort and Organize Data:**

Sorts Zip Codes by state and identifies **extreme geographic points** (north, south, east, west).

- **Generate CSV Headers:**

Adds appropriate headers to the output CSV files.

- **Produce CSV Output:**

Exports the sorted data to **CSV files**.

Key Components:

- **std::map<std::string, std::vector<ZipCodeRecord>> sortBuffer();**

Sorts Zip Code data and finds the **northernmost, southernmost, easternmost, and westernmost points** for each state.

- **void addHeader(std::string& file_name);**

Adds a **header row** to the specified CSV file.

- **bool csvOutput(std::string& file_name);**

Writes the **sorted data** to a CSV file.

Preliminary Documentation for CSVLengthIndicated.h and CSVLengthIndicated.cpp

Overview:

The **CSVLengthIndicated class** converts standard **CSV files** to **length-indicated format**, where each field is **prefixed by its length**. This format allows for efficient parsing of variable-length records.

Purpose:

- **Convert to Length-Indicated Format:**

Transforms CSV files into **length-indicated ASCII format**.

- **Ensure Data Consistency:**

Truncates fields exceeding the length limit and formats **floating-point numbers**.

Key Components:

- **void convertCSVToLengthIndicated(const std::string& csvFile, const std::string& outputFile);**

Converts a **CSV file** to **length-indicated format**.

- **std::vector<std::vector<std::string>> readLengthIndicatedRecord(const std::string& filename);**

Reads records from a **length-indicated file** and stores them in a vector.

Preliminary Documentation for IndexFile.h and IndexFile.cpp

Overview:

The **IndexFile class** generates **index files** that map Zip Codes to their **offsets** in length-indicated files. This enables **fast lookups** and efficient access to specific records.

Purpose:

- **Create Index Files:**

Maps Zip Codes to offsets, allowing **quick access** to records.

- **Handle Large Datasets:**

Optimizes file access through indexed lookups.

Key Components:

- **bool createIndexFile(const std::string& csvFile, const std::string& outputFile);**

Creates an **index file** from a length-indicated data file.

Preliminary Documentation for main.cpp

Overview:

The **main.cpp** file serves as the entry point for the application. It coordinates the usage of **Buffer**, **CSVProcessing**, **CSVLengthIndicated**, and **IndexFile** classes to read, process, and output Zip Code data.

Purpose:

- **Process and Sort CSV Files:**

Uses csvConvert_sort() to **generate CSV files** with headers and sorted data.

- **Convert to Length-Indicated Format:**

Converts CSV files to **length-indicated ASCII format**.

- **Generate Index Files:**

Creates index files for fast lookups using the **IndexFile class**.

Key Components:

- **void csvConvert_sort(CSVProcessing origin, std::string file);**

Adds a **header** to the CSV and outputs sorted data.

- **IndexFile::createIndexFile()**

Generates **index files** for fast lookups.