Custom Text Editor Overview

1. Features

This text editor is a command-line based, Vim-inspired editor implemented in C++. It supports both **normal** and **insert** modes, various navigation options, editing features, search and replace functionalities, and file handling commands.

Modes

- **Insert Mode (`i`)**: Allows typing text into the document.
- **Normal Mode (default)**: Used for navigation and command execution.

---

2. Controls and Commands

Insert Mode Controls

- `Arrow Keys` → Move cursor (Up: ↑ / 65, Down: ↓ / 66, Right: → / 67, Left: ← / 68)
- `Enter` → Insert a new line
- `Backspace` → Delete the character before the cursor
- `ESC` → Exit insert mode

Normal Mode Controls

Navigation:

- `Arrow Keys` or `65-68` → Move cursor (Up, Down, Right, Left)
- `w` → Move to next word
- `b` → Move to previous word
- `e` → Move to end of current word

- `0` → Move to start of line
- `$` → Move to end of line

Line Movement:

- `a` → Move down 5 lines
- `n` → Insert new line
- `J` → Join current and next line

Editing:

- `x` → Delete character at cursor
- `d` → Delete to end of line
- `l` → Delete two lines
- `k` → Delete three lines
- `z` → Delete a specific line (asks for line number)
- `f f` → Delete current line

Indentation:

- `>` → Indent current line
- `<` → Unindent current line

Copy/Paste:

- `y` → Yank (copy) current line
- `Y` → Yank (copy) two lines

- `p` → Paste after current line
- `P` → Paste before current line

Search and Replace:

- `/` → Search for a pattern in the current line
- `m` → Move to next match
- `N` → Move to previous match
- `;` → Search and replace command input (e.g. `:s/old/new/g`)

History & Info:

- `h` → Display command history

File Commands (triggered with `:`):

- `:w` → Save file
- `:q` → Quit
- `:wq` → Save and quit
- `:q!` → Force quit without saving

---

3. Data Structures Used

- **`std::vector<std::string>`**
  Stores all lines of the document, allowing random access and dynamic resizing. Each line is a separate string element.
- **`std::stack<std::string>` or `std::vector<std::string>` (for yank buffer)**
  Temporarily stores lines for `yank`, `paste`, and undo operations.

- **`std::string`**
  Used extensively for line content manipulation, character insertion, deletion, search, and replace.

- **`std::map`** / **`std::unordered_map`** *(if used in internal logic)*
  For storing command mappings or maintaining command history.

---

4. Utility Functions

- **`getChar()`**: Reads keyboard input, including handling of special keys (arrow keys via `_getch()`).

- **`handleFileCommand()`**: Parses and executes file-related commands starting with `:`.

- **`handleSearchReplaceCommand()`**: Processes search and replace functionalities.

---

5. Conclusion

This text editor emulates core features of Vim within a terminal environment using C++. It offers a mixture of mode-based editing, keyboard navigation, and text manipulation. The use of standard STL containers ensures efficient memory management and performance.