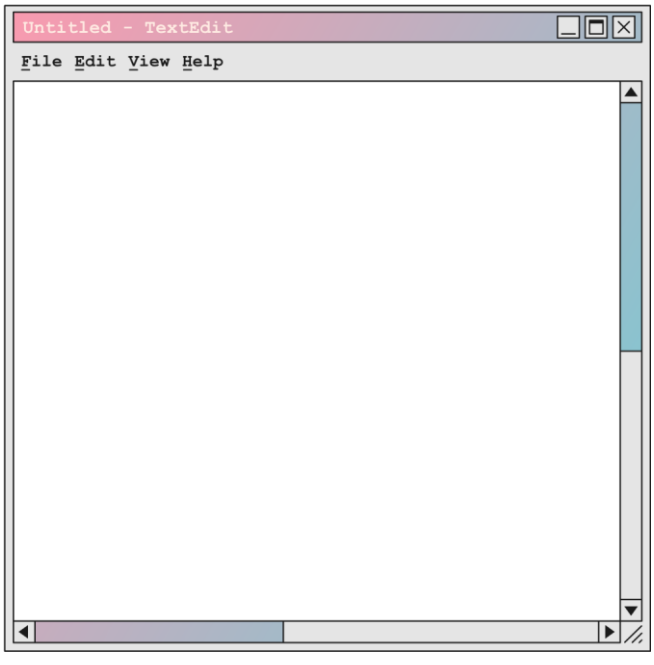
Report-03

Algorithm and Implementation

Submitted by
MERWIN PINTO
SRN 202100102
ROLL NO 1



Problem Statement:

Design and implement a basic text editor with fundamental features, such as text entry, editing, saving, and opening files. Include support for features like cut, copy, paste, undo, and redo. Develop a simple user interface for your editor and provide documentation on how to use these basic features effectively.

Algorithm and Data Strcuture:

Algorithm:

If we break down the code:

1.Initialization and GUI Setup:
The code starts by initializing a Tkinter application (self.root) to create the main window for the text editor.
The window's title is set to "Text Editor."

2.Toolbar Creation:
The program creates a toolbar at the top of the window (toolbar), which will hold various buttons for performing text editing operations.

3. Toolbar Buttons:
The following buttons are created on the toolbar:
"Cut": Calls the cut_text function.
"Copy": Calls the copy_text function.
"Paste": Calls the paste_text function.
"Undo": Calls the undo_text function.
"Redo": Calls the redo_text function.
"Open": Calls the open_file function.
"Save": Calls the save_file function.
These buttons are placed on the toolbar and serve as user interface elements for interacting with the text editor.

4. Text Widget Creation:
The program creates a Text widget (self.text_widget) in the main area of the window. This text widget is where the user can view and edit text.

5. History for Undo/Redo:
The code maintains a history of text changes to enable undo and redo functionality. The history is stored in a list (self.history), and the self.history_index keeps track of the current position in the history list.

6. Keyboard Shortcuts:
The code binds keyboard shortcuts to specific functions:
<Key>: Calls the on_text_change function whenever a key is pressed, recording text changes.
<Control-x>: Cuts the selected text and records the change.
<Control-c>: Copies the selected text.
<Control-v>: Pastes text from the clipboard and records the change.
<Control-z>: Initiates an undo operation.

7. Text Editing Functions:
The program defines various text editing functions that are associated with the toolbar buttons and keyboard shortcuts:
cut_text: Cuts the selected text and records the change in the history.
copy_text: Copies the selected text to the clipboard.
paste_text: Pastes text from the clipboard and records the change in the history.
undo_text: Reverts to a previous state in the history (undo).
redo_text: Moves forward in the history to redo actions.

8. Clipboard Operations:
Two functions are provided to manage the clipboard:
copy_to_clipboard: Clears the clipboard and appends the provided text.
get_clipboard_text: Retrieves text from the clipboard.

9. Recording Changes:
The record_change function captures the current text content and adds it to the history list.
It ensures that the history remains consistent with the user's actions.

10. Open and Save File:
The program allows the user to open and save text files:
open_file: Opens a file dialog to load the content of a selected file into the text editor.

save_file: Opens a file dialog for saving the current text to a file with a ".txt" extension.

11. Application Entry Point:
The program checks whether it's running as the main program (not imported as a module).
If it's the main program, it creates an instance of the TextEditorApp class and starts the main event loop using app.root.mainloop(). This is essential for the GUI to function.
In summary, this algorithm outlines the functionality and flow of a basic text editor with GUI features. The code's implementation utilizes the Tkinter library to create a graphical user interface for text editing and includes features like undo/redo, clipboard operations, and file open/save functionality.

Data Structure:

In this code, several data structures are used to implement the functionality of a text editor. Here are the main data structures used along with explanations:

1.Class:
TextEditorApp: This class serves as the main data structure that encapsulates the entire text editor application. It contains attributes and methods for managing the GUI, text editing, clipboard, and history.

2.Attributes within the TextEditorApp class:
self.root: The main Tkinter window, which acts as the container for the entire application.
self.text_widget: A Text widget within the window where the user can edit and view text.
self.history: A list that stores snapshots of the text content after each change, enabling undo and redo operations.
self.history_index: An integer representing the current position in the history list.

3.Local Variables within Methods:
Local variables are used within various methods, such as selected_text, clipboard_text, and file_path, to hold temporary data needed for specific operations.

4.Data Structures for GUI Elements:

tk.Frame and tk.Button: These Tkinter data structures are used for creating the toolbar and toolbar buttons. The toolbar holds buttons for various text editing operations.
tk.Text: The main text editing area is represented by a Text widget, which allows multi-line text input and display.

5.List:
self.history is a list used to store previous versions of the text content. It keeps a history of text changes to enable undo and redo functionality.
Dictionary (Not explicitly used in the code):

6.Dictionaries are not used in this code, but they are a common data structure for various applications. In a text editor, you might use a dictionary to store key-value pairs for settings or configurations.

7.String:
Strings are used extensively in the code to store text content. They represent the text displayed in the text widget, the content in the clipboard, and the file paths.

8.Integer:
The self.history_index variable is an integer used to keep track of the current position in the history list.
These are the primary data structures used in the code to create the text editor application, manage text content, and support various user interface elements and operations.

Conclusion:

In summary, the code is a basic text editor with a graphical interface. It mainly uses strings, lists, and integers to manage text content, history, and system programming is not a primary focus.