Deliverable 1

Problem Statement and Software Requirements

Group 4 — A CLI based Editor - Bloc++

Name	Github ID	Student ID
Nirmal Patel	lamrin13	101220783
Shivayashwanth Chikkondra	shivayashwanth	101193045

Github Repository: Group_4_Bloc++

Course Instructor: Professor Cristina Ruiz Martin

October 23, 2021

1 Software Design

This software is meant to allow users to create and/or update the program files, scripts or any other text files using the command line interface. To edit any file, user must run this software by passing the file name as the argument to this software. There are two possibilities with the given the filename,

- If the file exits, the bloc++ software will open this file with it's contents.
- If there is no such file with the given name and extensions, the software will create a new file in current working directory and open an empty file to edit.

After successfully opening a file, users are presented with two modes of operations, below is the brief description of the available features of each mode,

- **Command Mode:**
  - In command mode user can perform save operation, which asks the software to save changes made in the editor to the file system.
  - In command mode user has option to quit the editor, in this case if there are any unsaved changes the editor will warn the user about it. If they still want to quit without saving then the software won't save the changes and simply exit the editor and return to command prompt.
  - The user will be allowed to search for a keyword in the opened file. If the searched keyword is present in the file it will be highlighted with different color.
- **Insert Mode:** In this mode user will be allowed to write and or delete characters from the file. User can also used system defined copy and paste functionalities in this mode.

As this software is mainly for editing program files, it will have features such as highlighting the language specific keywords and numbers. As a basic feature the software will have syntax highlighting feature for only C language, but it can be extended for any other languages by adding list of keywords and extension in future. Moreover, for convenience of the programmer this software will auto close the brackets when the user opens any bracket.

After making changes in insert mode, user can switch to command mode and save changes. Finally, they can quit the editor and they will be returned to the command prompt to run the program.

2 Functions

This sections provide brief description of the functions with the input/output parameters and return type. The functions are color coded based on two releases, Gray for first release and Orange for the second (final release).

void raw_mode(*void*)
Assigned
Nirmal Patel
Description:
As we are going to launch are editor in command line itself, we need to block all the commands that triggers the programs built-in the command prompt, for example echo, grep, ls, etc. We need to block this commands in order to edit the files without calling this functions unnecessarily. This function will disable all such signal/functional calls to predefined programs of a shell and enter into raw mode in which user can type pretty much anything.
Results:
The command prompt will enter in raw mode, if the function is able to successfully block all the signals and commands to command prompt. On failure to do so the function will exit and the error message will be shown on command prompt.

void open_file (char *file_name)
Assigned
Nirmal Patel
Description:
This function will open a file by the given filename in the parameter, if the file already exists copy the file contents to the editor window, if not just open a empty file with the filename.
Parameters:
char *file_name: String value which specifies the name of the file to open.

void status_message (char *message)
Assigned
Shivayashwanth Chikkondra
Description:
This will be a helper function, which will be used by other functions to guide user or show available commands at the bottom of the screen. This function will print message string passed by the caller function at the bottom of the editor's screen.
Parameters:
char *message: String message value to be printed on the bottom of the editor window.

void process_keypress ()
Assigned
Nirmal Patel
Description:
This function will be called infinite times by main function which will listen to each key press and then process it accordingly.
Results:
If the editor is in insert mode and the key pressed is valid key then edit the contents of the editor window accordingly. If the editor is in command mode and key pressed is a command then call method assigned to that command.

void insert_newline ()
Assigned
Shivayashwanth Chikkondra
Description:
This function is called by process_keypress function when "Enter" key is pressed. And it will move the cursor to the next line ending the current line in the editor.
Results:
If the "Enter" key is pressed at the end of the content of the editor then it will simply move the cursor to the next line. But if the "Enter" key is pressed in the middle of the text content, then it will first move all the content below to one row down and create an empty row in between the content above.

void erase_character ()
Assigned
Shivayashwanth Chikkondra
Description:
This function is called by process_keypress function when Delete or Backspace key is pressed. And it will erase the character to the right or to the left of the cursor respectively.
Results:
If the backspace key is pressed charcter to the left of the cursor will be earased and the cursor will shift one character left. But if delete key is pressed then first the cursor is moved one step to the right and then performed backspace operation. Since this process is atomic the cursor movement is not observable.

void move_cursor (int key)
Assigned
Nirmal Patel
Description:
This function is called by process_keypress function when arrow keys are pressed. And it will move the cursor right, left, up or down accordingly.
Results:
If any of the arrow key is pressed the cursor's current position will change accordingly. But if the cursor is already at the bottom of the content and down key is pressed it would not move, same for the up arrow button and top of the editor. If the cursor is at the end of a line and right arrow key is pressed the cursor will move to the beginning of the next line. If the cursor is at the beginning of a line and left arrow key is pressed the cursor will move to the last character of the previous line.
Parameter:
int key: The ASCII code of the keys up, down, right and left arrow to be passed to the function to move cursor.

void save()
Assigned
Shivayashwanth Chikkondra
Description:
This function is called by process_keypress function when Ctrl+S key is pressed. It will prompt the software to save the contents of the file to file system.
Results:
A I/O stream is opened with the file name given at the beginning of the execution of the software. If the file already exist then replace the content of the file with the content of the editor, if file does not exist create a new file and write the content of editor to it and close the I/O stream. The variable to show that changes are saved is updated.

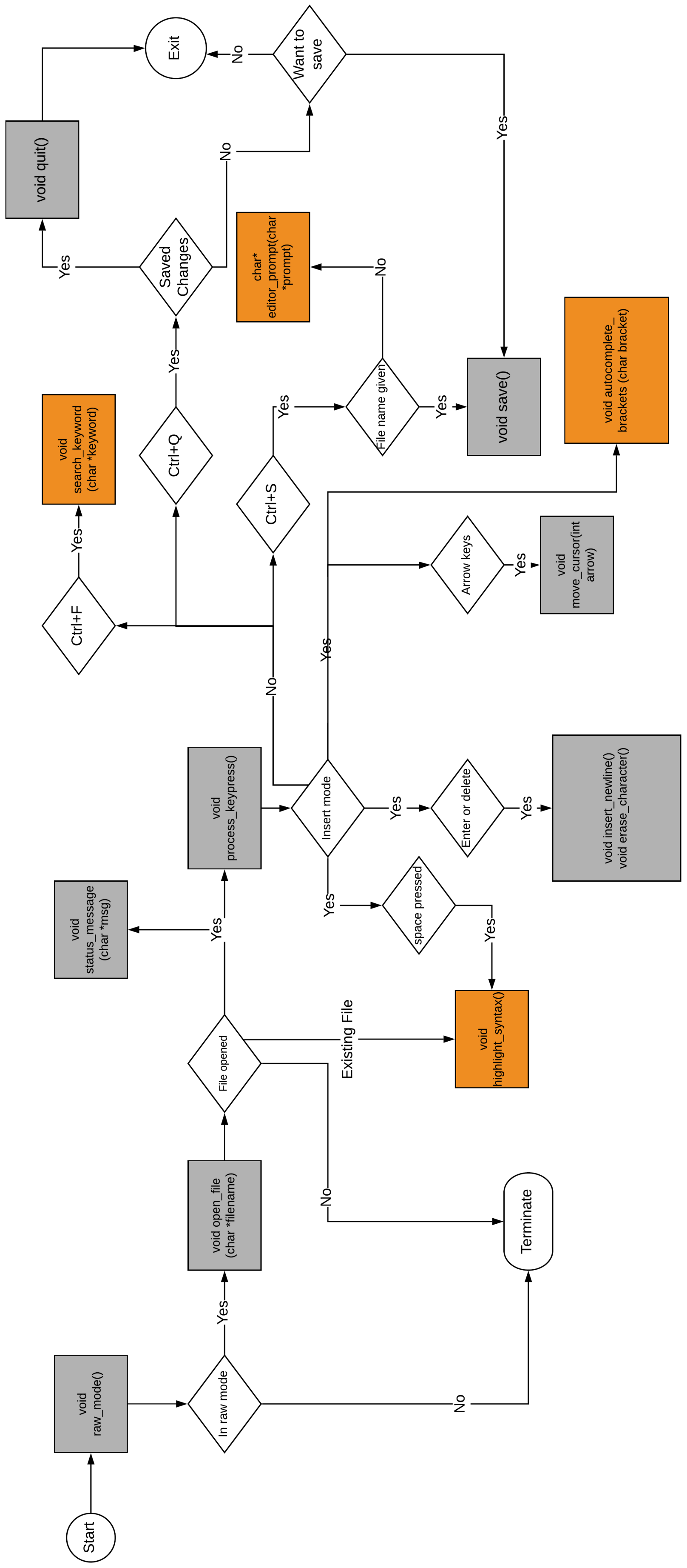
void quit()
Assigned
Nirmal Patel
Description:
This function is called by process_keypress function when Ctrl+Q key is pressed. It will prompt the software to close the editor and return the control to the command prompt.
Results:
If the changes to the file are already saved then close the editor and return to shell. But if changes are not saved, ask the user if they want to save or close. If they still choose to quit then close the editor without saving the chnages.

void highlight_syntax(char *language)
Assigned
Nirmal Patel
Description:
This function will highlight language keywords and numbers with different colors depending on the language provided by the user at the time of launching the editor by giving the extension to the file name. So, ".c" extensions will map to C language.
Results:
If user is opening a new file then this function will check for the keywords from the pre-defined list on every "space" key press. But if the user opens an existing file which already has some code then this function will go through each word and highlight the keywords accordingly.
Parameters:
char *language: The extension of the file based on which this function will highlight the keywords.

char* editor_prompt(char *promt)
Assigned
Shivayashwanth Chikkondra
Description:
This function will be called by other functions when the editor is in command mode when the software needs input from the user.
Results:
The prompt passed by the caller function will be displayed in the bottom of the screen and the user can respond accordingly. The response from the user will be returned to the caller function as a string.
Parameters:
char *prompt: Prompt text shown to the user to clarify what is expected from them.
Returns:
A string is returned, which has the input provided by the user.

void search_keyword(char *keyword)
Assigned
Shivayashwanth Chikkondra
Description:
This function will be called when Ctrl+F key is pressed in command mode. It will take an argument from the user as what keyword they want to search in the editor and search for that keyword in the editor from the top. If found it will highlight the word with different color.
Results:
The searched keyword will be highlighted with different color. If there are multiple words which matches the keyword then all the occurances will be highlighted.
Parameters:
char *keyword: String type keyword to be searched in the text.

void autocomplete_brackets(char bracket)
Assigned
Nirmal Patel
Description:
This function will be called whenever an open bracket key i.e., '(', '{', '[' and '<' is pressed and it will complete it's closing bracket.
Results:
The closing bracket of the inserted bracket will be automatically written on the editor and the cursor will be placed in between two brackets allowing user to write in between.
Parameters:
char bracket: Input character (one of the brackets).


Start
void raw_mode()
In raw mode
Yes
No
Terminate
void open_file (char *filename)
File opened
No
Existing File
void highlight_syntax()
Yes
void status_message (char *msg)
void process_keypress()
Insert mode
Yes
space pressed
Yes
Yes
Enter or delete
Yes
void insert_newline() void erase_character()
Yes
Arrow keys
Yes
void move_cursor(int arrow)
void autocomplete_ brackets (char bracket)
No
Ctrl+F
Yes
void search_keyword (char *keyword)
Ctrl+Q
Yes
Saved Changes
Yes
void quit()
No
Want to save
No
Exit
Yes
Ctrl+S
Yes
File name given
Yes
void save()
No
char* editor_prompt(char *prompt)