



Project Report on
COMMAND BASED TEXT EDITOR

Submitted by:
ANEESH PANCHAL - 2K20/MC/21
AYUSHI SAGAR - 2K20/MC/35

Submitted to:
PROF. GOONJAN JAIN

Department of Applied Mathematics
Delhi Technological University



Text Editor



Certificate

I hereby certify that the project dissertation titled “Command Based Text Editor” which is submitted by Aneesh Panchal (2K20/MC/21) and Ayushi Sagar (2K20/MC/35) of Mathematics and Computing Department, Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of the degree of Bachelor of Technology, is a record of the project work carried out by the students. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this university or elsewhere.

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042



Acknowledgement

We, Aneesh Panchal(2K20/MC/21) and Ayushi Sagar (2K20/MC/35) would like to express our special thanks of gratitude to Prof. Goonjan Jain, Mathematics and Computing Department, Delhi Technological University for his able guidance and support in completing our project report.

We came to know about many new things, we are really thankful to him. We would also like to thank our classmates who have also helped whenever we were stuck at some point.

Thanking You

Aneesh Panchal (2K20/MC/21)

Ayushi Sagar (2K20/MC/35)



Index

- Introduction
- Linked List
- Advantages of linked list
- Disadvantages of linked list
- Application of linkedList
- Application of linked list in real world
- Stacks
- Application of Stacks in data structure
- Application of stack in real world
- Command Based Text Editor
- The Codes and Files
- Description of Commands
- Description of Functions
- Conclusion and Summary of the code flow
- References





Introduction:

For the project we tried to built a command based text editor which mainly used two data structure that is linked list and stacks. The editor operates line by line i.e. we can insert a line,delete a line,save the .txt file,the autosave function is there and many more.

Talking in details , as soon as we enter inside the editor it asks us to open a file if it already exists or can create a new file also. After that, it asks us to whether we want to autosave our file or not. We provided a 'help' function which gives all the commands we can perform in the editor. After then it asks us to enter the command . Firstly , ofcourse we wanted to enter a line there is insert command there we can insert. We can perform many more such actions given all the details below.

Below is the execution of our code.

Some description of data structures we used in our project is provided by us.

```
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS E:\Codes> cd "e:\Codes\CS 251 DS\Command Based Text Editor Short Code\" ; if ($?) { g++ CBTE.cpp -o CBTE } ; if ($?) { .\CBTE }

Open or Create file:
create hello.txt

Do you want to autosave your progress ? (yes/no)
yes

Enter the Command:
show
Page empty !!!

Enter the Command:
insert 1 hello its me

Enter the Command:
insert 2 hello Ayushi

Enter the Command:
insert 3 hola

Enter the Command:
show
hello its me
hello Ayushi
hola

----- 3 lines -----

Enter the Command:
replace 3 hello Aneesh

Enter the Command:
show
hello its me
hello Ayushi
hello Aneesh

----- 3 lines -----

Enter the Command:
move 2 3

Enter the Command:
show
hello its me
hello Aneesh
hello Ayushi

----- 3 lines -----
```



```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
----- 3 lines -----
Enter the Command:
replace 3 hello Aneesh

Enter the Command:
show
hello its me
hello Ayushi
hello Aneesh
----- 3 lines -----
Enter the Command:
move 2 3

Enter the Command:
show
hello its me
hello Aneesh
hello Ayushi
----- 3 lines -----
Enter the Command:
undo

Enter the Command:
show
hello its me
hello Ayushi
hello Aneesh
----- 3 lines -----
Enter the Command:
delete 1

Enter the Command:
show
hello Ayushi
hello Aneesh
----- 2 lines -----
Enter the Command:
save

Enter the Command:
exit

PS E:\Codes\CS 251 DS\Command Based Text Editor Short Code>
```

```
E:\Codes\CS 251 DS\Command Based Text Editor\vx64(Debug)\CS251(DS).exe

CS 251 - Data Structures Project
Aneesh Panchal - 2K20/MC/21
Ayushi Sagar - 2K20/MC/35

Please Open or Create A.TXT File(ex : open 'Filename.txt' / create 'Filename.txt')
open test.txt

1 - This is the line 1
2 - This is the line 2
3 - This is the line 3
4 - This is the line 4
5 - This is the line 5
6 - This is the line 6
7 - This is the line 7
8 - This is the line 8
9 - This is the line 9
10 - This is the line 10

Current Page Number : 1 of 20 --> ( 200 lines )

Do you want the editor to autosave your progress ? (yes/no) :
yes

Enter Your Command ( to get command list, type help ) : help

Command List :

open          - To Open New File          ( ex : open test.txt )
next          - To Go Next Page
prev         - To Go Previous Page
insert {line no} {new text} - To Insert A New Line      ( ex : insert 5 MyNewText )
delete {line no} - To Delete A Line        ( ex : delete 5 )
replace {line no} {new text} - To Replace A Line Text    ( ex : replace 5 MyNewText )
move {line no1} {line no2} - To Move A Line To Target Line ( ex : move 5 7 )
save         - To Save Or Create A New File
undo         - To Undo Last Action
exit         - To Exit the Editor          ( Make sure to Save the file )

Enter Your Command ( to get command list, type help ) : _
```



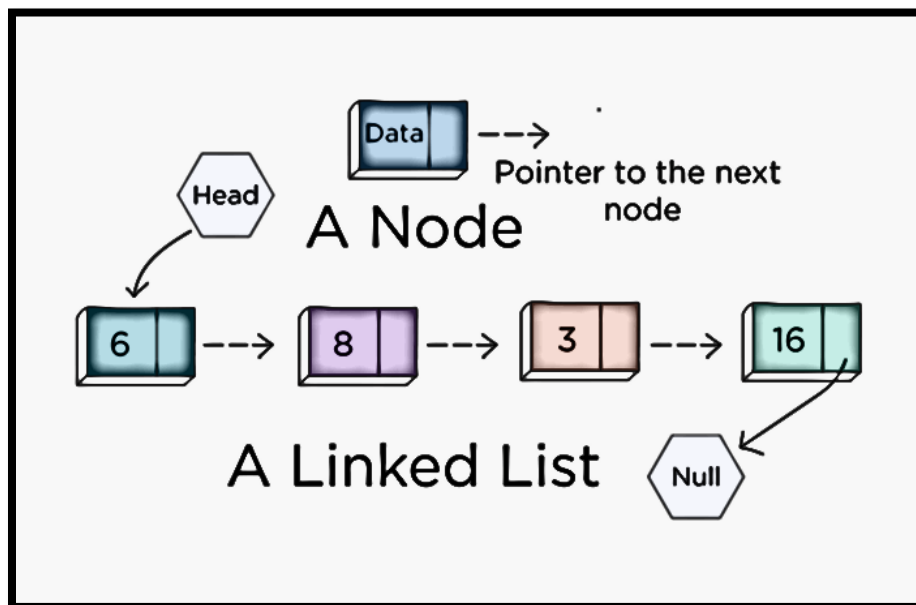
Linked list

Linked List is a very commonly used linear data structure which consists of group of nodes in a sequence.

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations

Each node holds its own data and the address of the next node hence forming a chain like structure.

Linked Lists are used to create trees and graphs.



Advantages of Linked Lists

- ❖ They are a dynamic in nature which allocates the memory when required.
- ❖ Insertion and deletion operations can be easily implemented.
- ❖ Stacks and queues can be easily executed.
- ❖ Linked List reduces the access time.

Disadvantages of Linked Lists

- ❖ The memory is wasted as pointers require extra memory for storage.
- ❖ No element can be accessed randomly; it has to access each node sequentially.
- ❖ Reverse Traversing is difficult in linked list.



Applications of Linked Lists

- ❖ Linked lists are used to implement stacks, queues, graphs, etc.
- ❖ Linked lists let you insert elements at the beginning and end of the list.
- ❖ In Linked Lists we don't need to know the size in advance.
- ❖ implementation of stack and queues
- ❖ implementation of graphs adjacency list representation of graphs is most popular which uses link list to store adjacent vertices
- ❖ dynamic memory allocation : we used list of free blocks
- ❖ maintain directory of names
- ❖ manipulation of polynomials by storing constants in the node of linked list
- ❖ representing sparse matrices

Application of linked list in real world

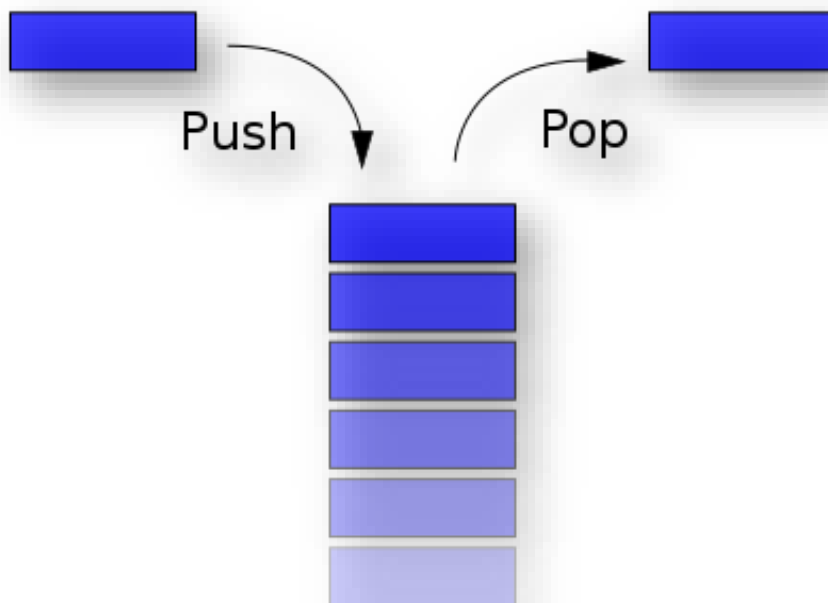
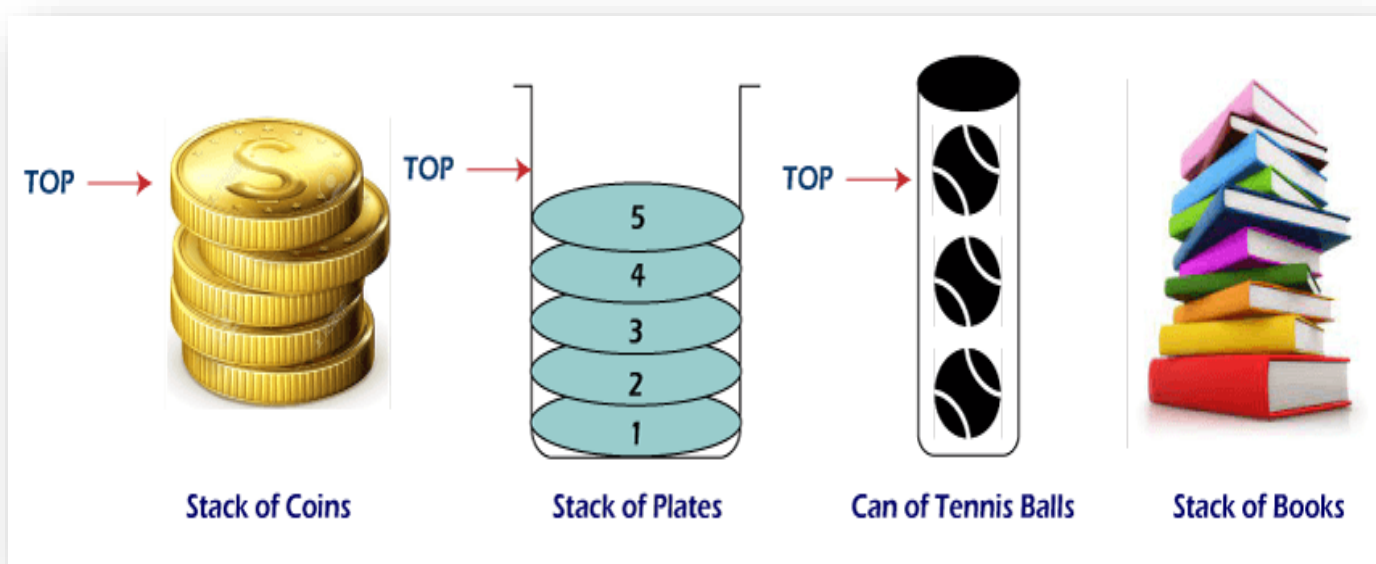
- ❖ image viewer - previous and next images are linked hence can be accessed by next and previous button
- ❖ web browser - previous and next url searched in web browser by pressing back and next button they are linked as linked list.
- ❖ music player - songs in music player are linked to previous and next song you can play songs either from starting or ending of the list



STACKS

A Stack is a widely used linear data structure in modern computers in which insertions and deletions of an element can occur only at one end, i.e., top of the Stack. It is used in all those applications in which data must be stored and retrieved in the last.

An everyday analogy of a stack data structure is a stack of books on a desk, Stack of plates, table tennis, Stack of bootless, Undo or Redo mechanism in the Text Editors, etc.





Applications of Stack in Data Structure:

- ❖ Evaluation of Arithmetic Expressions
- ❖ Backtracking
- ❖ Delimiter Checking
- ❖ Reverse a Data
- ❖ Processing Function Calls

Application of stack in real world

- ❖ To reverse a word. You push a given word to stack - letter by letter - and then pop letters from the stack.
- ❖ An "undo" mechanism in text editors; this operation is accomplished by keeping all text changes in a stack.
 - Undo/Redo stacks in Excel or Word.
- ❖ Language processing :
 - space for parameters and local variables is created internally using a stack.
 - compiler's syntax check for matching braces is implemented by using stack.
- ❖ A stack of plates/books in a cupboard.
- ❖ Wearing/Removing Bangles.
- ❖ Support for recursion
 - Activation records of method calls.



Stacks



TEXT EDITOR

Text editor we can say ability to change text by adding, deleting and rearranging letters, words, sentences and paragraphs. Text editing is the main operation users perform in word processors, which typically also handle graphics and other multimedia files.

A text editor is basically a type of computer program that edits plain text. Such programs are sometimes known as "notepad" software, following the naming of Microsoft Notepad. Text editors are provided with operating systems and software development packages, and can be used to change files such as configuration files, documentation files and programming language source code.

A text editor written or customized for a specific use can determine what the user is editing and assist the user, often by completing programming terms and showing tooltips with relevant documentation. Many text editors for software developers include source code syntax highlighting and automatic indentation to make programs easier to read and write. Programming editors often let the user select the name of an include file, function or variable, then jump to its definition. Some also allow for easy navigation back to the original section of code by storing the initial cursor location or by displaying the requested definition in a popup window or temporary buffer. Some editors implement this ability themselves, but often an auxiliary utility like ctags is used to locate the definitions.



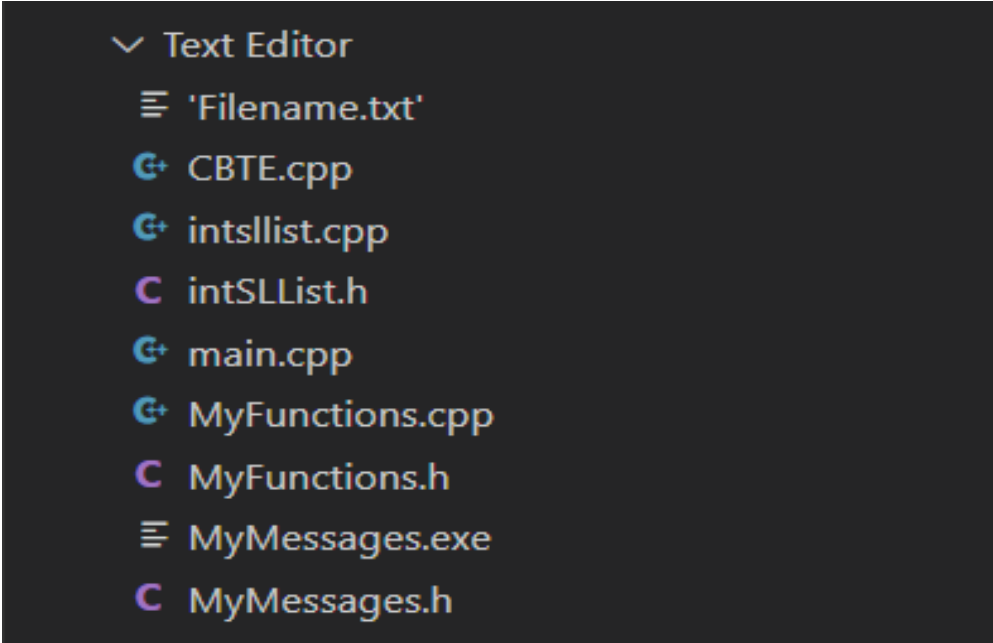


Our goal in this project is to write a command based text editor in c++ using linked list and stacks.

This text editor will print the contents of a file to the screen, page by page.

We have to implement a stack to keep track of all actions carried out, so you can undo actions if needed.

Sample text files are included to the folder of the project.



```
Text Editor
├── 'Filename.txt'
├── CBTE.cpp
├── intslldlist.cpp
├── intSLList.h
├── main.cpp
├── MyFunctions.cpp
├── MyFunctions.h
├── MyMessages.exe
└── MyMessages.h
```

❖ Header files

- **My.Messages.h** - it contains all the message we will display
- **MyFunctions.h** - it contains all the functons
- **intSLList.h** - it contains basic structure of the lists which we used

❖ Source files

- **MyFunctions.cpp** - it contains all the source codes for the functions which we define in functions.h
- **Intslldlist.cpp** - it contain functions related to Linked lists like insertion deletion

- ❖ **Main.cpp** - it contain the main driver code of the text editor



Main commands we have implimented:

```
E:\Codes\CS 251 DS\Command Based Text Editor\y64\Debug\CS251(DS).exe

CS 251 - Data Structures Project
Aneesh Panchal - 2K20/MC/21
Ayushi Sagar - 2K20/MC/35

Please Open or Create A.TXT File(ex : open 'Filename.txt' / create 'Filename.txt')
open test.txt

1 - This is the line 1
2 - This is the line 2
3 - This is the line 3
4 - This is the line 4
5 - This is the line 5
6 - This is the line 6
7 - This is the line 7
8 - This is the line 8
9 - This is the line 9
10 - This is the line 10

Current Page Number : 1 of 20 --> ( 200 lines )

Do you want the editor to autosave your progress ? (yes/no) :
yes

Enter Your Command ( to get command list, type help ) : help

Command List :

open          - To Open New File          ( ex : open test.txt )
next          - To Go Next Page
prev         - To Go Previous Page
insert {line no} {new text} - To Insert A New Line      ( ex : insert 5 MyNewText )
delete {line no} - To Delete A Line      ( ex : delete 5 )
replace {line no} {new text} - To Replace A Line Text    ( ex : replace 5 MyNewText )
move {line no1} {line no2} - To Move A Line To Target Line ( ex : move 5 7 )
save          - To Save Or Create A New File
undo         - To Undo Last Action
exit         - To Exit the Editor          ( Make sure to Save the file )

Enter Your Command ( to get command list, type help ) : _
```

Above are basically the commands we are can execute in our text editor like for opening or creating a file, in the file to insert or delete text or lines, jumping from one page to another by next and previous, we can even replace or move a text.

given below is a **Description of Commands** we are executing.

- ❖ **open filename** : It will open the file whose name is provided in the field filename and loads its contents. for example, if the user enters “openstest.text;”, our program should open test.txt and load each line in the file to the linked list.
- ❖ **save filename** : It will write the contents of the linked list to the file whose name is provided in the filename field.
- ❖ **delete n** : It will delete the line at position n.
- ❖ **move n m** : It will move the line at position n to new position m.



- ❖ **insert n text** : It will insert a new line with the "text" area user entered to the nth line. For example, if the user enters "insert 5 hello my friend", our program should insert a new line at line position 5 and put "hello my friend" into that line. If the .txt file already contains more than n lines, it should insert this new line between n-1 and n, putting the newly inserted text at line n. If the file contains less than n lines, it should fill the gap with blank lines until the newly inserted line becomes at position n.
- ❖ **replace n text** : It will replace the text at line n with the string provided within 'text' area.
- ❖ **next** : Previous contents of the file should not change, but the program should display the next page.
- ❖ **prev** : Previous contents of the file should not change, but the program should display the previous page.
- ❖ **help** : show all of the commands which user can use in this text editor
- ❖ **exit** : exit from the text editor (make sure you save the program before exit)
- ❖ **undo** : Reverts the last action taken. User should be able to call as many undo commands as s/he likes, and should be able to revert back to the initial state of the file after calling undo action enough times

Description of the function in MyFunction.cpp

- ❖ **LeftTrim** : It will remove the spaces from left side so if in cases some user enters spaces between two commands, our code will go on without interruption.
- ❖ **RightTrim** : It will remove the spaces from right side so if in cases some user enters spaces between two commands, our code will go on without interruption.



- ❖ **LeftAndRightTrimSpaces** : It will remove the spaces from left and right side so if in cases some user enter spaces between two commands , our code will go on without interruption.
- ❖ **ToLowerCase** : It will make the input string into lower case letter for easy comparison of strings and limiting the Case sensitive errors.
- ❖ **PageCountCalculator** : It will count the no. of pages in the text file.
- ❖ **SplitStringByDelimiterToArray** : Here we are using # char as delimiter, this will split the command easily into an array.

Description of functions in intsl1ist.cpp

- ❖ **addToHead** - It will insert the node at the head in linked list
- ❖ **addToTail** - It will insert the node at the tail in linked list
- ❖ **deleteFromHead** - It will delete the node from head in linked list
- ❖ **deleteFromTail** - It will delete the node from tail in linked list
- ❖ **deleteAll** - It will empty all the linked list
- ❖ **DeleteNode** – It will delete the specific node which is called
- ❖ **findNodeElementText** – It will return the text in the specific node which represents a line
- ❖ **printAll** – It will print total lines(total nodes) and number of pages



- ❖ **insertNode** – It will insert the node in the specific position
- ❖ **moveNode** - It will move the node from one position to another
- ❖ **replaceNode** - It will replace the node with given new node
- ❖ **Save** - It will save the file we will be working on.

Description of functions in CBTE.cpp

- ❖ **totalLine** - It will traverse the linked list and return the no. of Lines.
- ❖ **read** - It will read the file and make a linked list with its each line as a node
- ❖ **insert** - It will insert the node in the linked list
- ❖ **deleted** - It will delete the node in the linked list
- ❖ **replace** - It will replace the node with given new node
- ❖ **move** -It will move the node from one position to another
- ❖ **show** - It will print the node
- ❖ **save** - It will save the file we will be working on.
- ❖ **split** - We are using # char as delimiter, this will split the command easily into an array.



Conclusion and Summary of the Code flow:

We have as many functionalities as we can in this Command Based Text Editor. We have used it as a Command Based “Line” Text Editor as our Mid Term Evaluation Project.

Code Flow:

First of all we read the file line by line and we have created Linked List with data as string which will include the line as a string. Or we may create the file and then insert the data we want.

Then we have to empty the stack we have created. `Stack<string>`

Then we got the full linked list as 1st line as head node and 2nd line as head->next node. Then we can do any operation we want in this Linked List like **Insertion**, **Deletion**, **Replace** or **Move** Node positions as per user want to.

Also along with the operation we have implemented we also inserted the same operation command entered by the user with a delimiter as # so that we can use it afterwards for the undo function.

We used stacks for the Undo function because Undo just revert the last command. So as we have inserted the last command and then ask for Undo, it will redo the last command as last command string goes in stack will get first out so we can simply revert that command with least Time complexity $O(1)$.

Undo for Insertion is Deletion and vice versa.

Undo for move is move itself.

Undo for replace is also replace itself. (replace the text at given position)

Finally we have a Linked List after all the operations we have done then we can just save the file with file handling.

References:

- [Wikipedia.org](https://www.wikipedia.org)
- [Geeksforgeeks.org](https://www.geeksforgeeks.org)
- [Tutorialspoint.com](https://www.tutorialspoint.com)
- [Github.com](https://github.com)
- stackoverflow.com