**GROUP 7 : TEXT EDITOR IN C**

**DESCRIPTION :**

1. HEADER FILES :

#include <stdio.h>

#include <string.h>

This include standard C library header files.

* <stdio.h> is for input and output operations.
* <string.h> is for string manipulation functions.

1. MACROS :

#define MAX\_WORDS 100

#define MAX\_WORD\_LENGTH 100

These are preprocessor directives defining constants.

* MAX\_WORDS specifies the maximum number of words the text editor can handle
* MAX\_WORD\_LENGTH specifies the maximum length of each word.

1. GLOBAL VARIABLES :

char words[MAX\_WORDS][MAX\_WORD\_LENGTH];

int numWords = 0;

char clipboard[MAX\_WORD\_LENGTH];

* words is a two dimensional character array used to store words of text.
* numWords keeps track of the number of words in the editor.
* clipboard is used to store cut or copied words temporarily for pasting functionality.

1. FUNCTIONS :

The code contains several functions to perform various operations:

* displayMenu() : Displays the menu of options for the user.
* File Operations :

1. createNewFile() : Resets numWords to 0, creating a new empty file.
2. openFile() : Reads words from a file specified by the user and stores them in the words array.
3. saveFile() : Writes the content of words array to a file specified by the user.

* Text Manipulation Operations :

1. insertWord() : Inserts a new word at the specified position.
2. deleteWord() : Deletes a word at the specified position.
3. copyWord() : Copies a word to the clipboard.
4. cutWord() : Copies a word to the clipboard and deletes from the text.
5. pasteWord() : Pastes the word copied in the clipboard at the specified position.
6. printText() : Prints the entire text.
7. MAIN FUNCTION :

int main(){

\\Code

}

The main function presents a menu to the user and

processes user input in a loop until the user chooses to exit.

1. USER INPUT HANDLING :

The program uses :

* scanf : For user input handling.
* fopen, fprintf, fscanf : For file operations.

1. LOGIC AND FLOW :

* The program uses :

1. Loops
2. Conditional statements

To manage file operations, text manipulation and user input validation.

* The program also uses :

1. Switch statements

To perform different actions based on the user choice.

1. Function Calls

Based on the user's choice, specific functions are called to perform the corresponding operations.

1. ERROR HANDLING :

* The program includes basic error handling, like checking for file open errors, invalid user input and Maximum Word limit.

**C PROGRAM FOR TEXT EDITOR :**

#include <stdio.h>

#include <string.h>

#define MAX\_WORDS 100

#define MAX\_WORD\_LENGTH 100

char words[MAX\_WORDS][MAX\_WORD\_LENGTH];

int numWords = 0;

char clipboard[MAX\_WORD\_LENGTH];

void displayMenu() {

printf("\nText Editor\n");

printf("1. Create a new file\n");

printf("2. Open file\n");

printf("3. Insert\n");

printf("4. Delete\n");

printf("5. Copy\n");

printf("6. Cut\n");

printf("7. Paste\n");

printf("8. Save\n");

printf("9. Print text\n");

printf("10. Exit\n");

printf("Enter your choice: ");

}

void createNewFile() {

numWords = 0;

printf("New file created.\n");

}

void openFile() {

char fileName[MAX\_WORD\_LENGTH];

printf("Enter the file name to open: ");

scanf("%s", fileName);

FILE \*file = fopen(fileName, "r");

if (file == NULL) {

printf("Error opening the file.\n");

return;

}

numWords = 0;

while (numWords < MAX\_WORDS && fscanf(file, "%s", words[numWords]) != EOF) {

numWords++;

}

fclose(file);

printf("File opened successfully.\n");

}

void insertWord() {

int position;

printf("Enter position to insert: ");

scanf("%d", &position);

if (position < 1 || position > numWords + 1) {

printf("Invalid position.\n");

return;

}

if (numWords >= MAX\_WORDS) {

printf("Too many words. Cannot insert.\n");

return;

}

for (int i = numWords; i >= position; i--) {

strcpy(words[i], words[i - 1]);

}

printf("Enter word to insert: ");

scanf("%s", words[position - 1]);

numWords++;

printf("Text inserted successfully.\n");

}

void deleteWord() {

int position;

printf("Enter position to delete: ");

scanf("%d", &position);

if (position < 1 || position > numWords) {

printf("Invalid position.\n");

return;

}

for (int i = position - 1; i < numWords - 1; i++) {

strcpy(words[i], words[i + 1]);

}

numWords--;

printf("Text deleted successfully.\n");

}

void copyWord() {

int wordIndex;

printf("Enter word index to copy: ");

scanf("%d", &wordIndex);

if (wordIndex < 1 || wordIndex > numWords) {

printf("Invalid word index.\n");

return;

}

strcpy(clipboard, words[wordIndex - 1]);

printf("Text copied successfully.\n");

}

void cutWord() {

int wordIndex;

printf("Enter word index to cut: ");

scanf("%d", &wordIndex);

if (wordIndex < 1 || wordIndex > numWords) {

printf("Invalid word index.\n");

return;

}

strcpy(clipboard, words[wordIndex - 1]);

for (int i = wordIndex - 1; i < numWords - 1; i++) {

strcpy(words[i], words[i + 1]);

}

numWords--;

printf("Text cut successfully.\n");

}

void pasteWord() {

int position;

printf("Enter position to paste: ");

scanf("%d", &position);

if (position < 1 || position > numWords + 1) {

printf("Invalid position.\n");

return;

}

if (numWords >= MAX\_WORDS) {

printf("Too many words. Cannot paste.\n");

return;

}

for (int i = numWords; i >= position; i--) {

strcpy(words[i], words[i - 1]);

}

strcpy(words[position - 1], clipboard);

numWords++;

printf("Text pasted successfully.\n");

}

void saveFile() {

char fileName[MAX\_WORD\_LENGTH];

printf("Enter the file name to save: ");

scanf("%s", fileName);

FILE \*file = fopen(fileName, "w");

if (file == NULL) {

printf("Error opening the file.\n");

return;

}

for (int i = 0; i < numWords; ++i) {

fprintf(file, "%s ", words[i]);

}

fclose(file);

printf("File saved successfully.\n");

}

void printText(){

printf("Text in the editor:\n");

for (int i = 0; i < numWords; ++i) {

printf("%s ", words[i]);

}

}

int main() {

int choice;

while (1) {

displayMenu();

scanf("%d", &choice);

switch (choice) {

case 1:

createNewFile();

break;

case 2:

openFile();

break;

case 3:

insertWord();

break;

case 4:

deleteWord();

break;

case 5:

copyWord();

break;

case 6:

cutWord();

break;

case 7:

pasteWord();

break;

case 8:

saveFile();

break;

case 9:

printText();

break;

case 10:

printf("Exiting the text editor.\n");

return 0;

default:

printf("Invalid choice. Please try again.\n");

break;

}

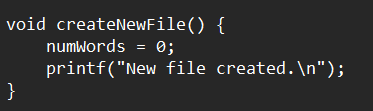
}

return 0;

}

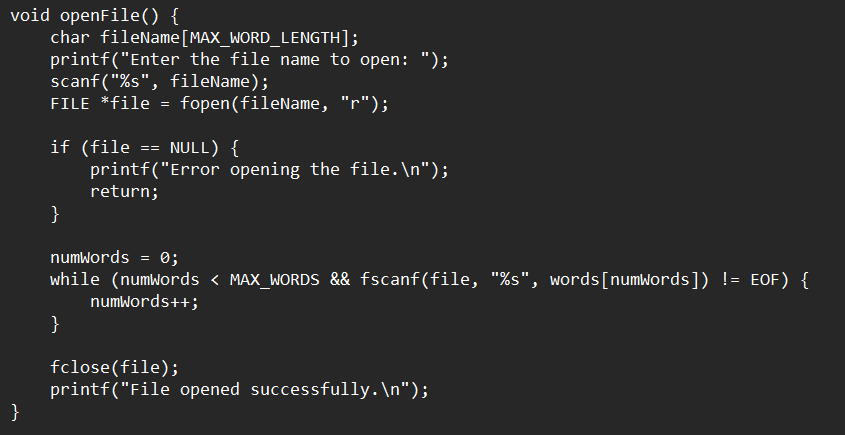
**EXPLANATION:**

1. Create a New File :



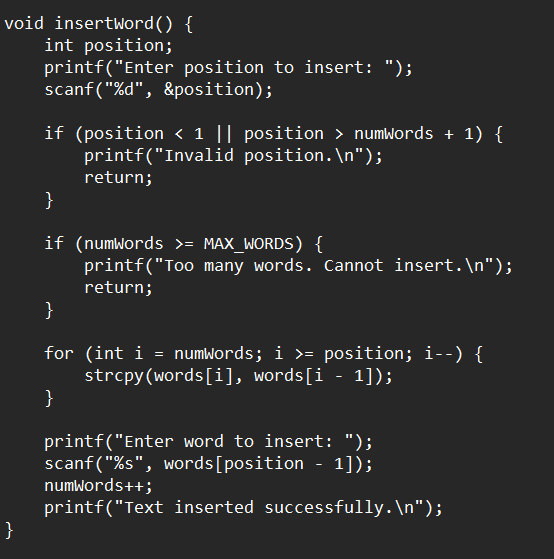
* numWords = 0; : Setting numWords to 0 signifies that a new, empty file is being created.
* printf("New file created.\n"); : This line displays the message "New file created."

1. Open File :



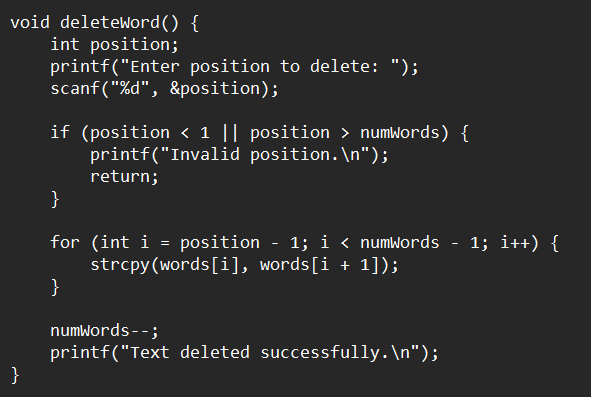
* The function starts by asking the user to enter the name of the file they want to open.
* It declares a character array fileName to store the user input for the file name.
* The fopen function is then used to attempt to open the specified file in read-only mode ("r").
* If the file doesn't exists, fopen returns NULL, indicating an error in opening the file.
* If the file is successfully opened, the function initializes numWords to 0, preparing the array to store new words.
* It then enters a loop to read words from the file using fscanf.
* fscanf reads words from the file and stores them in the words array.
* The loop continues until it fills up MAX\_WORDS or reaches the end of the file (EOF).
* After reading words from the file or encountering an error, the file is closed using the fclose function to free up system resources.
* Finally, the function prints a message indicating whether the file was successfully opened or if there was an error.

1. Insert Text :



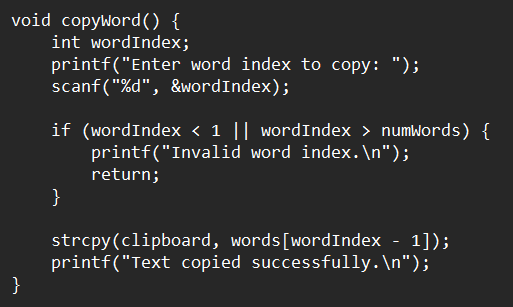
* The function starts by asking the user to enter the position at which they want to insert a new word.
* If the position < 1 or position > numWords + 1, an error message is displayed for invalid position. The function then exits.
* The function checks if there is enough space in the words array to insert a new word. If numWords >= MAX\_WORDS, it indicates that there are too many words, and the function displays an error message before exiting.
* If the input position is valid and there is space, the function shifts existing words in the words array to the right to make room for the new word.
* This is achieved by iterating from the last word to the position where the new word needs to be inserted and copying each word to the next index.
* After making space, the function prompts the user to enter the new word that they want to insert at the specified position.
* The new word is stored in the words array at the correct position.
* The numWords variable is incremented to reflect the addition of the new word.
* Finally, the function prints a message indicating that the text was inserted successfully.

1. Delete Text :



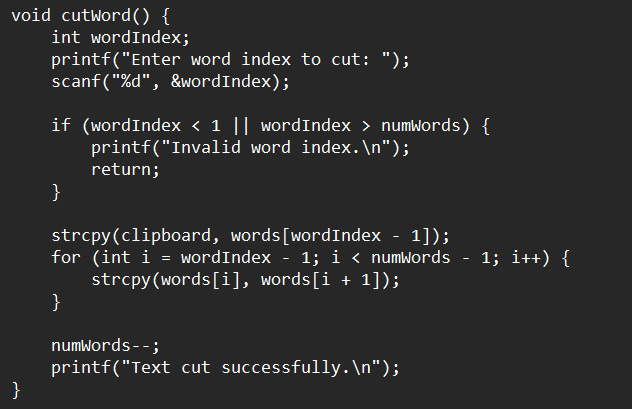
* The function starts by entering the position of the word the user wants to delete.
* If the position < 1 or position > numWords, it's considered an invalid position, and an error message is displayed.
* If the input position is valid, the function shifts the words in the words array to the left, effectively removing the word at the specified position.
* The loop starts from the index of the word to be deleted and continues until the second-to-last word in the array.
* Each word is replaced by the word in the next index.
* After shifting the words, numWords is decremented to reflect the removal of a word.
* The function prints a message indicating that the text was deleted successfully.

1. Copy Text :



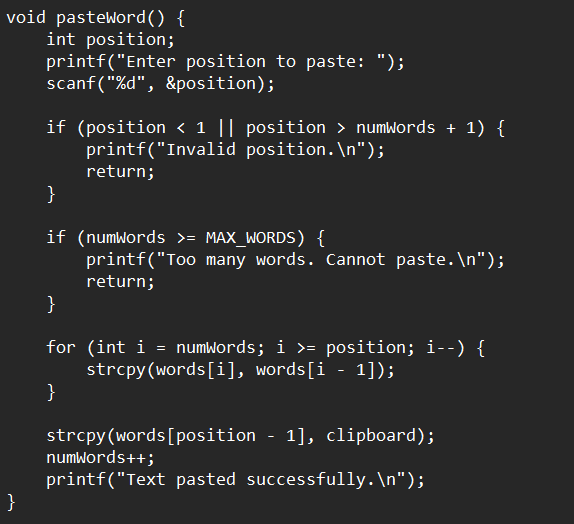
* The function starts by asking the user to enter the index of the word they want to copy.
* If the wordIndex < 1 or wordIndex > numWords, it's considered an invalid index, and an error message is displayed. The function then exits.
* If the input index is valid, the function copies the word at the specified index from the words array to the clipboard array.
* The strcpy() function is used to perform this copy operation.
* Finally, the function prints a message indicating that the text was copied successfully.

1. Cut Text :



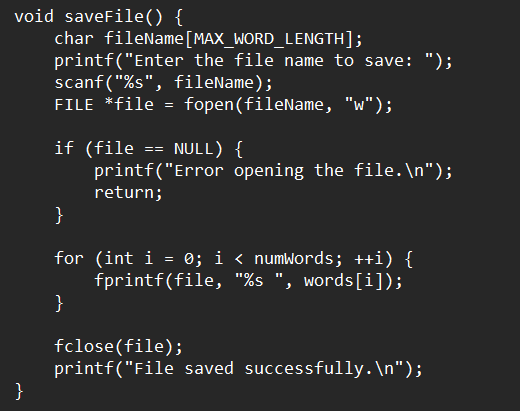
* The function starts by asking the user to enter the index of the word they want to cut.
* If wordIndex < 1 or wordIndex > numWords(current number of words), it's considered an invalid index, and an error message is displayed. The function then exits.
* If the input index is valid, the function copies the word at the specified index from the words array to the clipboard array.
* The strcpy() function is used to perform this copy operation.
* After copying the word to the clipboard, the function shifts the remaining words in the words array to fill the gap left by the cut word. This ensures there are no empty spaces in the array.
* A loop iterates through the array, starting from the index where the word was cut, and each word is replaced by the next word in the array.
* The variable numWords is decremented by 1 to reflect the removal of the word from the array.
* Finally, the function prints a message indicating that the text was cut successfully.

1. Paste Text :



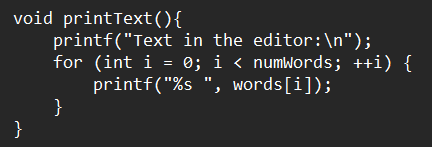
* The function starts by asking the user to enter the position where they want to paste the text.
* If the position < 1 or position > numWords + 1, an error message is displayed.
* The function checks if there is enough space in the words array to accommodate the pasted word. If the numWords >= MAX\_WORDS, the array is full, and the function displays an error message before exiting.
* If the input position is valid and there is space in the array, the function shifts the existing words in the words array to make room for the pasted word.
* A loop iterates through the array, starting from the last word, moving each word one position to the right, to create a space for the new word.
* After creating the necessary space, the function copies the word stored in the clipboard array into the specified position in the words array.
* The variable numWords is incremented by 1 to reflect the addition of the pasted word.
* Finally, the function prints a message indicating that the text was pasted successfully.

1. Save Text :



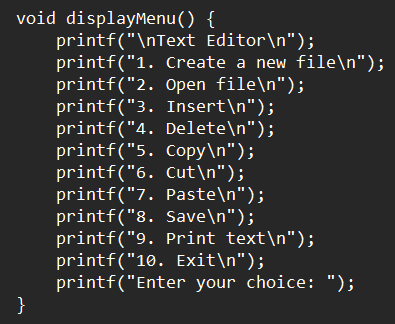
* The function begins by asking the user to enter the name of the file they want to save.
* The function then attempts to open the specified file in write mode ("w"). If the file does not exist, it will be created. If the file opening fails, an error message is displayed.
* If the file is successfully opened, the function enters a loop that iterates through the words array. For each word, it uses fprintf() to write the word followed by a space into the file. This creates a space-separated list of words in the file.
* After writing all words to the file, the function closes the file using fclose() to ensure that all changes are saved to the file.
* If the file was saved successfully, the function prints a message that the file was saved successfully.

1. Print Text :

****

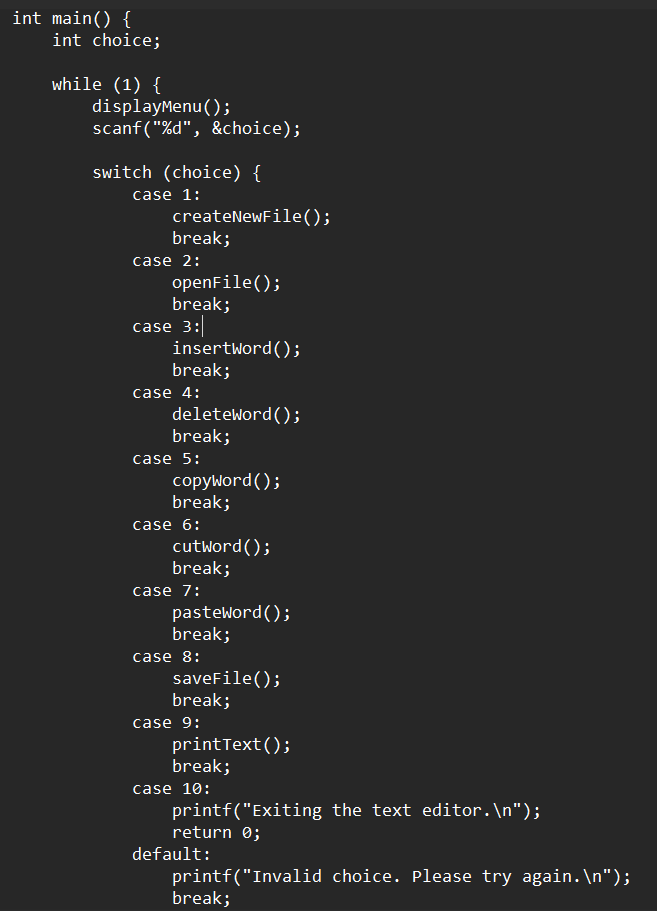
* The function enters a loop that iterates through the words array. It uses a for loop with the loop variable i starting from 0 and ending at numWords - 1.
* Within the loop, the function prints each word in the words array followed by a space. The %s format specifier is used to print strings in C.
* After printing all words, the loop moves to the next line by printing a newline character (\n). This ensures that the next output appears on a new line.

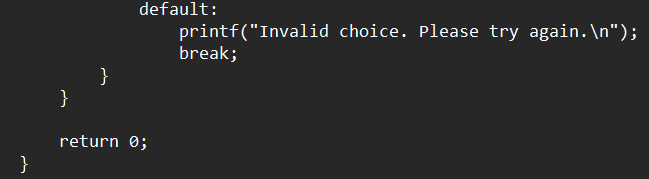
1. Display Menu :



* The function prints the text "Text Editor" followed by the numbered menu options. Each printf statement prints a specific menu option followed by a newline character (\n) to move to the next line.
* The function prints the message "Enter your choice: " to prompt the user to input their choice.
* The function does not have a return statement, as it simply prints the menu and waits for user input. Once the function execution reaches the closing brace '}', it completes.

1. Main Function :

****

****

* The function starts by declaring an integer variable choice to store the user's menu choice.
* The while(1) loop creates an infinite loop, which means the program will keep running until it encounters a return statement.
* The displayMenu() function is called to print the menu options, allowing the user to choose an operation.
* The scanf() function is used to take the user's choice as input and store it in the choice variable.
* The switch statement is used to perform different actions based on the user's choice. Depending on the value of choice, the program will execute the corresponding case.
* Based on the user's choice, the program calls the corresponding function (e.g., createNewFile(), openFile(), insertWord(), etc.) to perform the selected operation.
* If the user chooses option 10, the program prints an exit message and returns 0, indicating successful termination of the program.
* If the user enters an invalid choice, the program prints an error message, prompting the user to try again.

**OUTPUT :**

|  | 1)Displays Menu |
| --- | --- |
|  | 2)New File Created |
|  | 3) Inserting text |
|  | 4)Saving  Filename: group |
|  | 5) Opening file ‘group’ |
|  | 6)Printing Text from the file |
|  | 7)Deleting a word from the file  Deleted word: everyone |
|  | 8)Reinserting the deleted word with “World!” |
|  | 9)Copying a word from file”project”    Copied word: consists |
|  | 10)Pasting the word in the file “group” |
|  | 11)Using Cut function to cut word “consists ”from file “group” |
|  | 12)Pasting the word “consists” in file group at another position. |
|  | 13)Exiting the text editor |