****

**Project**

***“*TEXT EDITOR*”***

Course Title: COAL

## Submitted To

Dr. Adeel Akram

## Submitted By

**Omaima** Roll no: SP23-BCS-013

**Aroona** Roll no: SP23-BCS-015

**Roomah** Roll no: SP23-BCS-102

*Date of submission:* 20-12-202

**CODE:**

. MODEL SMALL

. DATA

matrix db 80\*25 dup(?),'$' ;25 lines of 80 chars each (2000 bytes)

matrix\_2 db 22 dup(?)

row db 2 ;for navigating columns/character cells with arrow keys

column db 0 ;for navigating rows/lines with arrow keys

curr\_line db 2 ;rows/lines while editing

curr\_char db 0 ;columns/chars while editing

;for main menu

deco1 db ' =================================================$'

deco2 db '|| Command Line Text Editor ||$'

deco3 db '|| ||$'

deco4 db '|| ESC = Exit || CTRL+S = Save File ||$'

deco5 db '|| ARROW KEYS = Navigate ||$'

deco6 db ' =================================================$'

docPrompt db 'Enter Document Name (.txt): $'

docName dw 50 dup(?),'$'

openPrompt db 'Enter Document Name to Open: $'

HANDLE dw ?

header db 80 dup('='),'$'

color db 3\*15+15

filename db 'MYFILE.TXT$', 0

message db 'This is a sample text file.$'

. CODE

START:

; Initialize Data Segment

MOV AX, DATA

MOV DS, AX

; Create a file - DOS interrupt 21H, AH=3Ch

MOV AH, 3Ch ; Function to create a file

MOV CX, 0 ; File attributes (normal file)

MOV DX, OFFSET filename ; Pointer to the filename

INT 21H ; Call interrupt

MOV BX, AX ; File handle is returned in AX, move it to BX

; Write to the file - DOS interrupt 21H, AH=40h

MOV AH, 40h ; Function to write to a file

MOV CX, 27 ; Number of bytes to write (length of message)

MOV DX, OFFSET message ; Pointer to the message

INT 21H ; Call interrupt

; Close the file - DOS interrupt 21H, AH=3Eh

MOV AH, 3Eh ; Function to close a file

INT 21H ; Call interrupt

; Exit program

MOV AH, 4Ch ; Function to exit program

INT 21H

; =========== MACROS ===========

newline macro

mov dl, 10 ;newline ASCII

mov ah, 2

int 21h

mov dl, 13 ;linefeed (return) ASCII

mov ah, 2

int 21h

endm

remove macro

mov dx, 8 ;backspace to go back one char

mov ah, 2

int 21h

mov dx, 32 ;space to remove that char

mov ah, 2

int 21h

mov dx, 8 ;backspace to go back at removed char position

mov ah, 2

int 21h

endm

goto\_pos macro row, col

mov ah, 02h ;set text position in middle screen

mov dh, row

mov dl, col

int 10h

endm

clrScrn macro

mov ah, 02h ;set cursor to upper left corner

mov dh, 0

mov dl, 0

int 10h

mov ah, 0Ah ;overwrite with blank chars & remove all chars

mov al, 00h ;character

mov cx, 2000 ;how many times to write

int 10h ;graphics interrupt

endm

debug macro arg

mov dx, arg ;for debugging purpose

mov ah, 2

int 21h

endm

;=============== PROCEDURES ===============

start\_menu proc

;DISPLAY MAIN MENU

goto\_pos 5, 12

mov dx, offset deco1 ;decoration 1

mov ah, 9

int 21h

goto\_pos 6, 12

mov dx, offset deco2 ;decoration 2

mov ah, 9

int 21h

goto\_pos 7, 12

mov dx, offset deco3 ;decoration 3

mov ah, 9

int 21h

goto\_pos 8, 12

mov dx, offset deco4 ;decoration 4

mov ah, 9

int 21h

goto\_pos 9, 12

mov dx, offset deco5 ;decoration 5

mov ah, 9

int 21h

goto\_pos 10, 12

mov dx, offset deco6 ;decoration 6

mov ah, 9

int 21h

goto\_pos 13, 12

mov dx, offset docPrompt ;prompt doc name field

mov ah, 9

int 21h

;INPUT CHARS IN DOC NAME FIELD

mov cx, 0 ;array size counter

mov si, offset docName

input\_char:

mov ah, 1

int 21h

cmp al, 13 ;check if return key hit

je return

cmp al, 8 ;check if backspace key hit

je remove\_char

inc cx ;increment array size by 1

mov [si], al

inc si

jmp input\_char

remove\_char:

cmp cx, 0

je setPos\_ret

dec cx ;decrement array size by 1

dec si

mov [si], 00h

mov dl, 32 ;for removing char

mov ah, 2

int 21h

mov dl, 8

mov ah, 2

int 21h

jmp input\_char

setPos\_ret:

goto\_pos 13, 40

jmp input\_char

return: ;clear the screen and return procedure

ret

start\_menu endp

upper\_bar proc

goto\_pos 0 0

mov dx, offset docName ;display DOCNAME on upper corner

mov ah, 9

int 21h

goto\_pos 1 0

mov dx, offset header

mov ah, 9

int 21h

ret

upper\_bar endp

;=============== MAIN ===============

MAIN PROC

mov ax, @DATA

mov ds, ax

mov ah, 01h ;Define Text Cursor Shape

mov cx, 07h

int 10h

clrScrn

call start\_menu ;call start menu

clrScrn ;clear screen macro

call upper\_bar ;call upper stats bar in editor UI

goto\_pos 2, 0 ;set cursor position beneath upper bar

mov si, offset matrix

mov di, offset matrix\_2

MAIN\_LOOP:

; Get keystroke

mov ah, 00h

int 16h

; AH = BIOS scan code

cmp ah, 01h ;if escape key

je EXIT

cmp al, 13h ;if CTRL+S

je SAVE

cmp al, 0Fh ;if CTRL+O

je OPEN

cmp ah, 48h ;if up arrow

je UP

cmp ah, 50h ;if down arrow

je DOWN

cmp ah, 4Bh ;if left arrow

je LEFT

cmp ah, 4Dh ;if right arrow

je RIGHT

cmp ah, 1Ch ;if enter (newline) key

je ENTER

cmp ah, 0Eh ;if backspace (remove character)

je BACKSPACE

cmp column, 79

je ENTER

mov dl, al ;if any other key, then write char on screen

mov ah, 2

int 21h

mov [si], al ;add char in matrix array

inc si

inc curr\_char ;increment char position on current row

inc column ;also increment the current character count

goto\_pos row, column

jmp MAIN\_LOOP

EXIT:

mov ah, 4ch

int 21h

SAVE:

mov ah, 3Ch ;creating a file

mov cx, 0 ;read-only file

mov dx, offset docName ;giving name which we took from Main Menu Doc Name Input

int 21h

mov ah, 3Dh ;opening file

mov al, 1 ;for writing mode

mov dx, offset docName ;which file

int 21h

mov HANDLE, ax ;setting up handler

mov ah, 40h ;function for writing files

mov bx, HANDLE ;search for file handler

mov cx, 2000 ;how many bytes to write in file

mov dx, offset matrix ;what to write

int 21h

jmp MAIN\_LOOP

OPEN:

goto\_pos 22 0 ;go to bottom to write open file prompt

mov dx, offset openPrompt

mov ah, 9

int 21h

;INPUT CHARS IN DOC NAME FIELD

mov cx, 0 ;array size counter

mov di, offset docName

input\_char2:

mov ah, 1

int 21h

cmp al, 13 ;check if return key hit

je return2

cmp al, 8 ;check if backspace key hit

je remove\_char2

inc cx ;increment array size by 1

mov [di], al

inc di

jmp input\_char2

remove\_char2:

cmp cx, 0

je setPos\_ret2

dec cx ;decrement array size by 1

dec di

mov [di], 00h

mov dl, 32 ;for removing char

mov ah, 2

int 21h

mov dl, 8

mov ah, 2

int 21h

jmp input\_char2

setPos\_ret2:

goto\_pos 22, 29

jmp input\_char2

return2: ;clear the screen and return procedure

clrScrn

call upper\_bar

goto\_pos 2, 0 ;set cursor position beneath upper bar

mov ah, 0x3d ;to open files

mov al, 00 ;file handler for reading files

mov dx, offset docName

int 21h

mov HANDLE, ax ;setting up handler

mov ah, 0x3f ;function for reading files

mov bx, HANDLE

mov cx, 1760 ;how many bytes to write

mov dx, offset matrix ;where to save read data

int 21h

mov dx, offset matrix ;print the text on editor canvas

mov ah, 9

int 21h

jmp MAIN\_LOOP

UP:

cmp row, 2

je MAIN\_LOOP

dec curr\_line

dec row

goto\_pos row, column

jmp MAIN\_LOOP

DOWN:

inc curr\_line

inc row

goto\_pos row, column

jmp MAIN\_LOOP

LEFT:

dec column

goto\_pos row, column

jmp MAIN\_LOOP

RIGHT:

inc column

goto\_pos row, column

jmp MAIN\_LOOP

ENTER:

newline ;newline macro

mov [si], 10 ;move newline into array

inc si

mov dl, curr\_char

mov [di], dl

inc di

inc curr\_line

mov curr\_char, 0

inc row ;increment row number

mov column, 0 ;get hold of 0th Col for Navigation

goto\_pos row, 0 ;to get on newline 0th column

jmp MAIN\_LOOP

BACKSPACE:

;IF TRUE

cmp curr\_line, 2 ;see if cursor is on the very 1st line of document

;THEN DO THIS

je rmv ;if TRUE, then just Remove the chars from matrix

;IF TRUE

cmp curr\_char, 0 ;see if cursor is on the 0th POS on most left

;THEN DO THIS

je goBackLine ;if TRUE, then go back to upper row at the latest character's POS

;ELSE DO THIS

remove

dec curr\_char

dec column

dec si

mov [si], 00h

jmp MAIN\_LOOP

rmv:

remove

dec curr\_char

dec column

dec si ;decrement si

mov [si], 00h ;fill NULL in removed char space in array

jmp MAIN\_LOOP

goBackLine:

dec curr\_line

dec row

dec di

mov dl, [di]

mov column, dl

goto\_pos curr\_line, dl ;go to the last character position in previous row

mov dl, [di] ;moving in another register because size doesn't match

mov curr\_char, dl ;to reset the cursor to the last position of previous line

jmp MAIN\_LOOP

MAIN ENDP

END MAIN