Министерство образования Республики Беларусь

Учреждение образования Белорусский государственный университет информатики и радиоэлектроники

Кафедра ЭВМ

Отчёт по лабораторной работе №5

“Работа с файлами”

Проверил: Выполнил:

ассистент каф. ЭВМ студент гр.030501

Туровец Николай Олегович Вайтехович П.В.

Минск 2022

**Цель работы:** Ознакомиться с основными операциями обработки файлов, получить понятие о работе с параметрами командной строки.

**Алгоритм**

* Открытие существующего файла
* Выполнение файловых операций чтения и записи
* Закрытие файла

Для работы с файлами в данной лабораторной работе лучше всего использовать функции DOS, которые обращаются к файлу через 16-битный идентификатор (дескриптор) файла. Такой подход более прост, чем использование более старых описателей файла (37-байтного блока управления файлом FCB) или функций низкого уровня доступа к диску (прерывание BIOS 13h). Во всех случаях имя файла (если диск или путь отсутствуют в описании, то системой используются их текущие значения) описывается ASCIZ-строкой. Во всех случаях имя файла (если диск или путь отсутствуют в описании, то системой используются их текущие значения) описывается ASCIZ-строкой.

При запуске программы DOS помещает всю командную строку (включая последний символ 0Dh) в блок PSP запущенной программы по смещению 81h и ее длину в байт 80h. Длина командной строки, хранящейся в PSP, не может быть больше 126 символов (командная строка большей длины доступна начиная с DOS 4.0 в переменной среды CMDLINE).

При загрузке программы, в начале отводимого для нее блока памяти, создается структура данных PSP (префикс программного сегмента) размером 256 байт (100h). Затем DOS создает копию текущего окружения для загружаемой программы, помещает полный путь и имя программы в конец окружения, заполняет поля PSP, сама программа записывается в память, начиная с адреса PSP:0100h.

**Листинг программы**

.model small

.stack 100h

.data

filename db "filename.txt"

buffer db 128 dup(0)

buf db 0

handle dw 0

counter dw 0

c dw 0

flag db 0

space\_counter dw 0

closeString db "Close the file$"

errorExeString db "Atata, ne zapuskay .exe!$"

openFileError db "Error of open!$"

openString db "Open the file$"

newLine db 13, 10, '$'

errorString db "Error!$"

exitString db "Exit$"

lastSymbol db 0

.code

outputString proc

mov ah, 09h

int 21h

ret

outputString endp

printNewLine proc

lea dx, newLine

call outputString

ret

printNewLine endp

get\_name proc

push ax

push cx

push di

push si

xor cx, cx

mov cl, es:[80h]

cmp cl, 0

je end\_get\_name

mov di, 82h

lea si, filename

cicle1:

mov al, es:[di]

cmp al, 0Dh

je end\_get\_name

mov [si], al

inc di

inc si

jmp cicle1

end\_get\_name:

dec si

cmp BYTE PTR [si], 'e'

je exeError

cmp BYTE PTR [si], 'm'

je exeError

pop si

pop di

pop cx

pop ax

ret

get\_name endp

;Îòêðûòèå ôàéëà äëÿ ÷òåíèÿ è çàïèñè

fopen proc

mov ah, 3dh

mov al, 2

lea dx, filename

int 21h

jc openError

mov handle,

ret

fopen endp

;Çàêðûòèå ôàéëà

fclose proc

mov ah, 3eh

mov bx, handle

int 21h

jc error

ret

fclose endp

checkTab:

cmp BYTE PTR [si], 9

jne notWhiteSpace

jmp next

;Óäàëåíèå ïóñòûõ ñòðîê

proc space

mov counter, 0

mov space\_counter, 0

i:

mov cx, 128

mov bx, handle

lea dx, buffer

mov ah, 3fh

int 21h

jc error

xor cx, cx

mov cx, ax

jcxz close

push ax

xor si, si

mov c, 0

mov flag, 0

lea si, buffer

cmp BYTE PTR [si], 0

je close

k:

inc c

cmp BYTE PTR [si], 10

je endOfLine

cmp BYTE PTR [si], ' '

jne checkTab

next:

pop ax

cmp ax, c

je endOfLine

push ax

inc si

jmp k

jmp i

notWhiteSpace:

cmp BYTE PTR [si], 13

je cret

pop ax

cmp ax, c

je endOfLine

push ax

mov flag, 1

inc si

jmp k

nonEmpty:

;ïåðåìåùàåì óêàçàòåëü

xor ax, ax

mov bx, handle

mov ah, 42h

mov dx, counter

xor cx, cx

int 21h

xor ax, ax

mov bx, handle

mov ah, 40h

mov dx, offset buffer

xor cx, cx

mov cx, c

int 21h

mov ax, counter

add ax, c

mov counter, ax

mov ax, counter

add ax, space\_counter

mov counter, ax

xor ax, ax

mov bx, handle

mov ah, 42h

mov dx, counter

xor cx, cx

int 21h

mov ax, counter

sub ax, space\_counter

mov counter, ax

jmp i

Empty:

mov ax, c

add space\_counter, ax

mov ax, counter

add ax, space\_counter

mov counter, ax

xor ax, ax

mov bx, handle

mov ah, 42h

mov dx, counter

xor cx, cx

int 21h

mov ax, counter

sub ax, space\_counter

mov counter, ax

jmp i

endOfLine:

cmp flag, 1

je nonEmpty

jne Empty

cret:

pop ax

cmp ax, c

je endOfLine

push ax

inc si

jmp k

endp

error:

lea dx, errorString

call outputString

call printNewLine

jmp exit

exeError:

lea dx, errorExeString

call outputString

call printNewLine

jmp exit

openError:

lea dx, openFileError

call outputString

call printNewLine

jmp exit

begin:

mov ax, @data

mov ds, ax

call get\_name

call fopen

lea dx, openString

call outputString

call printNewLine

call space

jmp close

close:

xor ax, ax

mov bx, handle

mov ah, 42h

dec counter

mov dx, counter

xor cx, cx

int 21h

mov bx, handle

mov ah, 40h

int 21h

call fclose

lea dx, closeString

call outputString

call printNewLine

exit:

lea dx, exitString

call outputString

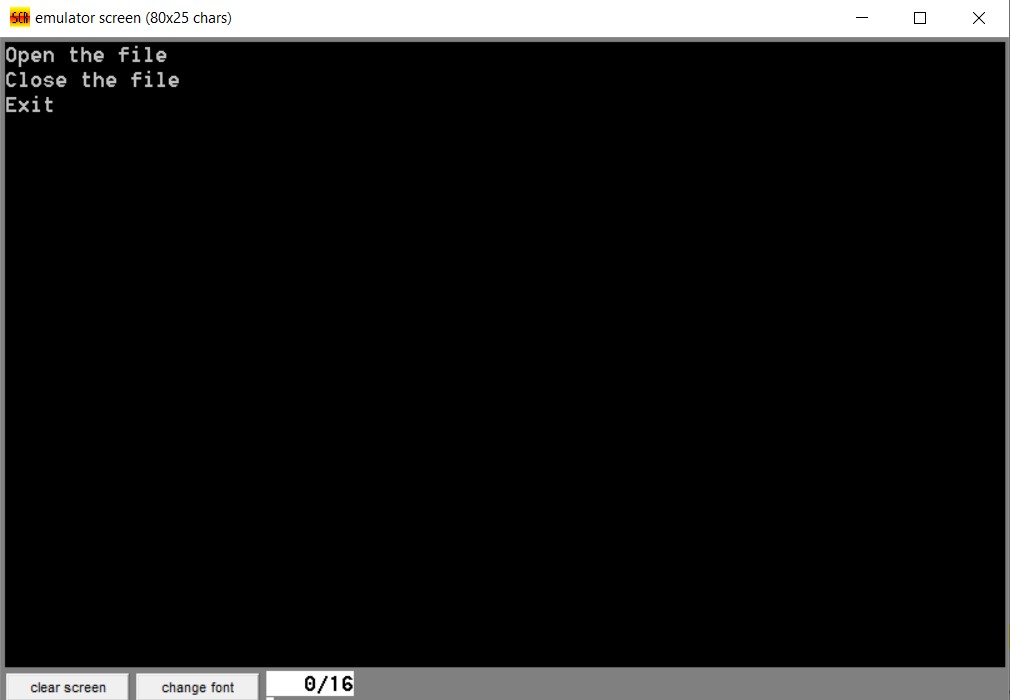
call printNewLine

mov ah, 4ch

int 21h

end begin

**ТЕСТ**

****

*Рисунок 1 –* результат выполнения программы

**Вывод:** В данной лабораторной работе разработана программа удаления всех неинформативных (пустых строк).