МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

# Кафедра проектирования информационно-компьютерных систем

Отчет по предмету

Программирование на языке ассемблера

Лабораторная работа №5

**«РАБОТА С ФАЙЛАМИ»**

|  |  |
| --- | --- |
| **Выполнил:**  Студент группы 150501  Смоленский Н.О. | **Проверил:**  Туровец Н.О. |

Минск 2022

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

**Теоретические сведения:**

1. *Работа с файлами.*

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).

Первые пять значений идентификаторов такого формата инициализируются системой следующим образом:

-- 0 – STDIN – стандартное устройство ввода (клавиатура),

-- 1 – STDOUT – стандартное устройство вывода (экран),

-- 2 – STDERR – устройство вывода сообщений об ошибках (всегда экран),

-- 3 – AUX – последовательный порт (СОМ1),

-- 4 – PRN – параллельный порт (LPT1).

Работа с файлами выполняется через функции DOS в стандартном порядке:

1) создание или открытие существующего файла;

2) выполнение файловых операций чтения или записи данных;

3) закрытие файла.

Дополнительно доступны операции: удаление, поиск и управление.

Для создания или открытия существующего файла рекомендуется ис-  
пользовать функции:

-- Функция DOS 3Ch (INT 21h) – создать файл;

-- Функция DOS 5Bh (INT 21h) – создать и открыть файл;

-- Функция DOS 5Ah (INT 21h) – создать и открыть временный файл;

-- Функция DOS 3Dh (INT 21h) – открыть существующий файл.

Во всех случаях имя файла (если диск или путь отсутствуют в описании,  
то системой используются их текущие значения) описывается ASCIZ-строкой  
(строкой ASCII-символов, оканчивающейся нулем), которая, например, имеет  
следующий вид:

filename db 'с:\data\filename.txt',0

Для выполнения файловых операций чтения или записи данных рекомендуется использовать функции:

-- Функция DOS 3Fh (INT 21h) – чтение из файла или устройства;

-- Функция DOS 42h (INT 21h) – переместить указатель чтения/записи;

-- Функция DOS 40h (INT 21h) – запись в файл или устройство;

-- Функция DOS 68h (INT 21h) – сброс файловых буферов DOS на диск;

-- Функция DOS 0Dh (INT 21h) – сброс всех файловых буферов на диск;

Для закрытия файла рекомендуется использовать следующую функцию:

-- Функция DOS 3Eh (INT 21h) – закрыть файл.

Дополнительные операции с файловой системой:

-- Функция DOS 41h (INT 21h) – удалить файл;

-- Функция DOS 4Eh (INT 21h) – найти первый файл;

-- Функция DOS 1Ah (INT 21h) – установить область DTA;

-- Функция DOS 4Fh (INT 21h) – найти следующий файл;

-- Функция DOS 39h (INT 21h) – создать каталог;

-- Функция DOS 3Ah (INT 21h) – удалить каталог;

-- Функция DOS 47h (INT 21h) – определить текущий каталог;

-- Функция DOS 3Bh (INT 21h) – сменить каталог.

2. *Работа с командной 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.

**Код программы:**

*.model small*

*.stack 100h*

*.data*

*file\_id dw 0*

*ptrh dw ?*

*ptrl dw ?*

*str\_length dw 0*

*num dw 0*

*flag db 0*

*msg1 db "Unable to open a file",0Dh,0Ah,'$'*

*msg2 db "Unable to close a file",0Dh,0Ah,'$'*

*msg3 db "Invalid input",0Dh,0Ah,'$'*

*success db "Deleted successfully!",0Dh,0Ah,'$'*

*command\_line db 126 dup(0)*

*file\_name db 125 dup(0)*

*buffer db ?*

*.code*

*printStr MACRO str*

*push ax dx*

*mov dx,offset str*

*mov ah,9*

*int 21h*

*pop dx ax*

*ENDM*

*openFile PROC*

*push cx dx*

*mov ah,3Dh*

*mov al,02h*

*mov cl,0*

*int 21h*

*jnc file\_opened*

*printStr msg1*

*file\_opened:*

*pop dx cx*

*ret*

*openFile ENDP*

*isDelimiter PROC*

*xor ah,ah*

*cmp al,20h*

*jne not\_space*

*mov ah,1*

*not\_space:*

*cmp al,9h*

*jne not\_tab*

*mov ah,1*

*not\_tab:*

*ret*

*isDelimiter ENDP*

*findWord PROC*

*push bp*

*mov bp,sp*

*push ax cx si*

*mov si,[bp+4]*

*handle\_word:*

*skip\_delimiter:*

*mov al,[si]*

*call isDelimiter*

*cmp ah,1*

*je found\_delimiter*

*jmp stop\_reading\_string*

*found\_delimiter:*

*cmp cx,0*

*jne string\_end\_not\_reached*

*jmp stop\_reading\_string*

*string\_end\_not\_reached:*

*inc si*

*dec cx*

*jmp handle\_word*

*stop\_reading\_string:*

*mov bx,si*

*sub bx,[bp+4]*

*skip\_character:*

*mov al,[si]*

*call isDelimiter*

*cmp ah,1*

*jne not\_delimiter*

*jmp word\_read*

*not\_delimiter:*

*cmp cx,0*

*jne can\_read\_more*

*jmp word\_read*

*can\_read\_more:*

*cmp al,0Dh*

*jne not\_the\_end*

*jmp word\_read*

*not\_the\_end:*

*inc si*

*dec cx*

*jmp skip\_character*

*word\_read:*

*mov di,si*

*sub di,[bp+4]*

*cmp bx,di*

*je word\_not\_found*

*dec dx*

*word\_not\_found:*

*cmp cx,0*

*jne not\_end\_of\_string*

*jmp end\_of\_loop*

*not\_end\_of\_string:*

*cmp dx,0*

*jne not\_this\_word*

*jmp end\_of\_loop*

*not\_this\_word:*

*cmp byte ptr [si],0Dh*

*jne not\_stop*

*jmp end\_of\_loop*

*not\_stop:*

*jmp handle\_word*

*end\_of\_loop:*

*pop si cx ax bp*

*ret*

*findWord ENDP*

*deleteStrings PROC*

*push ax bx cx dx*

*mov bx,file\_id*

*mov ah,42h*

*mov al,0*

*xor cx,cx*

*xor dx,dx*

*int 21h*

*deleteStr\_loop:*

*mov cx,num*

*cmp cx,0*

*je deleteStr\_delete*

*deleteStr\_skip:*

*call skipString*

*cmp flag,1*

*je deleteStr\_end*

*loop deleteStr\_skip*

*deleteStr\_delete:*

*call getLength*

*call deleteFromFile*

*cmp flag,1*

*je deleteStr\_end*

*jmp deleteStr\_loop*

*deleteStr\_end:*

*pop dx cx bx ax*

*ret*

*deleteStrings ENDP*

*skipString PROC*

*push ax dx cx*

*skipStr\_loop:*

*mov ah,3Fh*

*mov cx,1*

*mov dx,offset buffer*

*int 21h*

*cmp ax,0*

*je skipStr\_EOF*

*cmp [buffer],0Ah*

*je skipStr\_end*

*jmp skipStr\_loop*

*skipStr\_EOF:*

*mov flag,1*

*skipStr\_end:*

*pop cx dx ax*

*ret*

*skipString ENDP*

*getLength PROC*

*push ax dx cx*

*mov ah, 42h*

*mov al, 01h*

*xor cx, cx*

*xor dx, dx*

*int 21h*

*push ax dx*

*mov str\_length,0*

*length\_loop:*

*inc str\_length*

*mov ah,3Fh*

*mov cx,1*

*mov dx,offset buffer*

*int 21h*

*cmp ax,0*

*je length\_EOF*

*cmp [buffer],0Ah*

*je length\_end*

*jmp length\_loop*

*length\_EOF:*

*mov flag,1*

*length\_end:*

*mov ah,42h*

*mov al,00h*

*pop cx*

*pop dx*

*int 21h*

*pop cx dx ax*

*ret*

*getLength ENDP*

*deleteFromFile proc*

*push ax dx cx di*

*mov di,str\_length*

*mov ah,42h*

*mov al,01h*

*xor cx,cx*

*xor dx,dx*

*int 21h*

*mov ptrh,dx*

*mov ptrl,ax*

*deleteFromFile\_loop:*

*mov ah,42h*

*mov al,01h*

*xor cx,cx*

*xor dx,dx*

*int 21h*

*push ax*

*push dx*

*mov ah,42h*

*mov al,01h*

*xor cx,cx*

*mov dx,di*

*int 21h*

*mov ah,3Fh*

*mov cx,1*

*mov dx,offset buffer*

*int 21h*

*cmp ax, 0*

*je deleteFromFile\_EOF*

*mov ah,42h*

*mov al,00h*

*pop cx*

*pop dx*

*int 21h*

*mov ah,40h*

*mov cx,1*

*mov dx,offset buffer*

*int 21h*

*jmp deleteFromFile\_loop*

*deleteFromFile\_EOF:*

*mov ah,42h*

*mov al,00h*

*pop cx*

*pop dx*

*int 21h*

*mov ah,40h*

*mov cx,0*

*mov dx,offset buffer*

*int 21h*

*deleteFromFile\_end:*

*mov ah,42h*

*mov al,00h*

*mov cx,ptrh*

*mov dx,ptrl*

*int 21h*

*pop di cx dx ax*

*ret*

*deleteFromFile endp*

*readCommandLine PROC*

*push cx si di*

*xor cx,cx*

*mov cl,ds:[0080h]*

*mov bx,cx*

*mov si,81h*

*lea di,command\_line*

*rep movsb*

*pop di si cx*

*ret*

*readCommandLine ENDP*

*getFileNameFromCmd PROC*

*push cx bx dx si di*

*xor ax,ax*

*mov dx,1*

*mov cx,bx*

*push offset command\_line*

*call findWord*

*add sp,2*

*cmp dx,0*

*jne file\_name\_not\_found*

*mov cx,di*

*sub cx,bx*

*mov ax,cx*

*mov di,offset file\_name*

*mov si,bx*

*add si,offset command\_line*

*rep movsb*

*file\_name\_not\_found:*

*pop di si dx bx cx*

*ret*

*getFileNameFromCmd ENDP*

*isDigit PROC*

*xor ch,ch*

*cmp cl,30h*

*jge greater\_than\_zero*

*jmp not\_a\_number*

*greater\_than\_zero:*

*cmp cl,39h*

*jle less\_than\_nine*

*jmp not\_a\_number*

*less\_than\_nine:*

*mov ch,1*

*not\_a\_number:*

*ret*

*isDigit ENDP*

*getNumberFromCmd PROC*

*push bx cx dx si*

*mov dx,2*

*mov cx,bx*

*push offset command\_line*

*call findWord*

*add sp,2*

*cmp dx,0*

*jne invalid\_input*

*mov si,10*

*xor ax,ax*

*get\_another\_digit:*

*mov cl,command\_line[bx]*

*cmp bx,di*

*jne not\_end\_of\_number*

*jmp end\_of\_number*

*not\_end\_of\_number:*

*call isDigit*

*cmp ch,1*

*je digit\_read*

*jmp invalid\_input*

*digit\_read:*

*mul si*

*jno not\_overflow*

*jmp invalid\_input*

*not\_overflow:*

*xor ch,ch*

*sub cx,30h*

*add ax,cx*

*jno not\_of*

*jmp invalid\_input*

*not\_of:*

*inc bx*

*jmp get\_another\_digit*

*end\_of\_number:*

*jmp proc\_end*

*invalid\_input:*

*xor ax,ax*

*proc\_end:*

*pop si dx cx bx*

*ret*

*getNumberFromCmd ENDP*

*closeFile PROC*

*push ax*

*mov ah,3Eh*

*int 21h*

*jnc file\_closed*

*printStr msg2*

*file\_closed:*

*pop ax*

*ret*

*closeFile ENDP*

*start:*

*mov ax,@data*

*mov es,ax*

*call readCommandLine*

*mov ds,ax*

*call getFileNameFromCmd*

*cmp ax,0*

*je invalid\_parameter*

*call getNumberFromCmd*

*mov bx,ax*

*cmp ax,0*

*je invalid\_parameter*

*mov dx,offset file\_name*

*call openFile*

*jc \_end*

*mov file\_id,ax*

*mov num,bx*

*dec num*

*call deleteStrings*

*mov bx,file\_id*

*call closeFile*

*jc \_end*

*printStr success*

*\_end:*

*jmp skip\_output\_msg*

*invalid\_parameter:*

*printStr msg3*

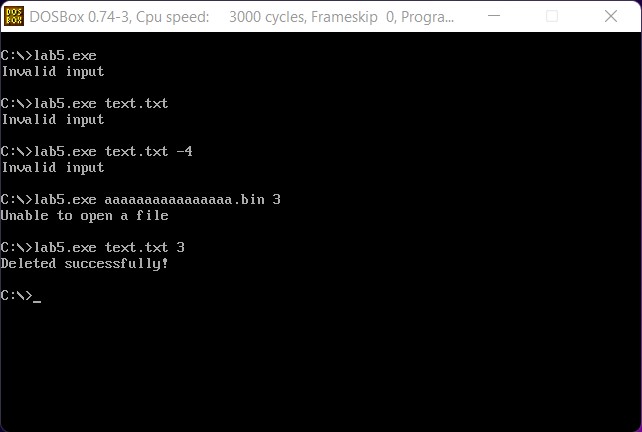
*skip\_output\_msg:*

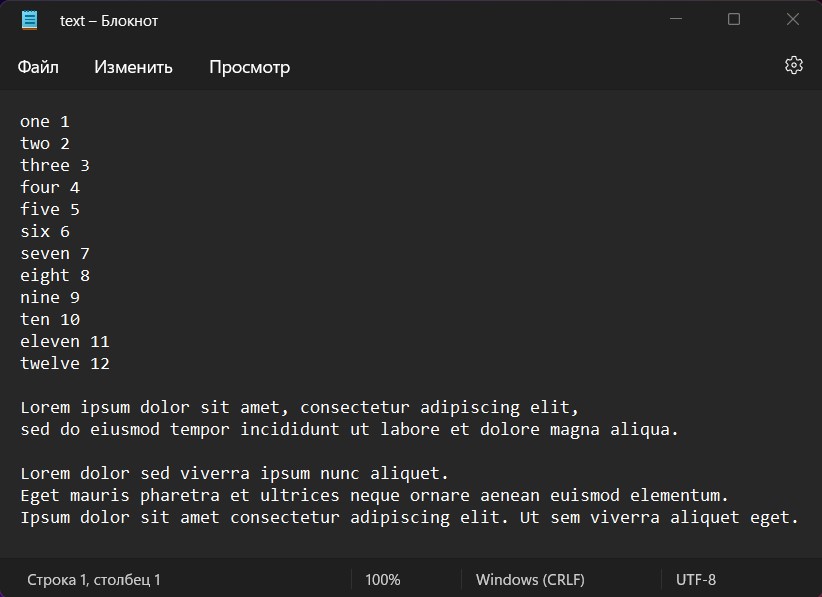
*mov ax,4c00h*

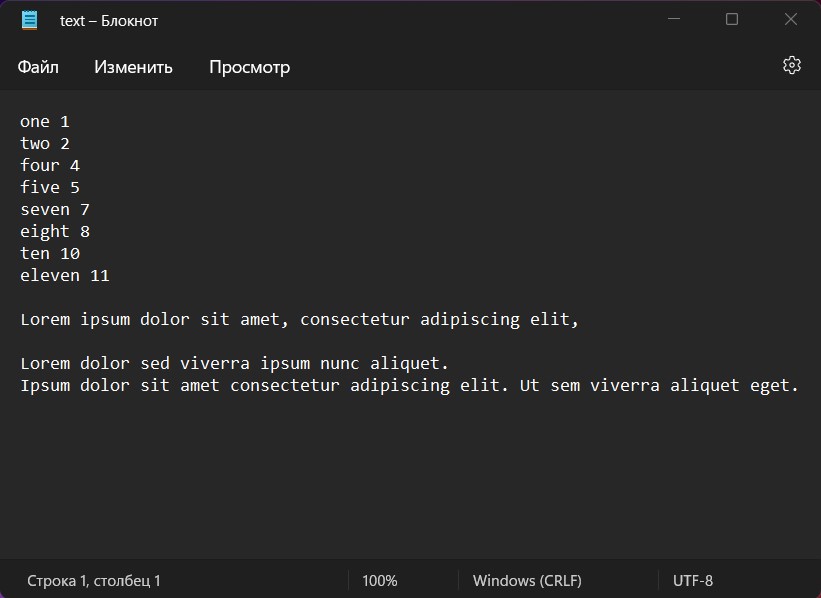
*int 21h*

*end start*

**Результат работы:**

****

****

****

**Вывод**

В ходе выполнения лабораторной работы было проведено ознакомление с основными операциями обработки файлов и работой с командной строкой.