Лабораторная работа № 10

Цепочечные команды. Работа с файлами.

Цель работы

Реализовать основные операции работы с файлами: открытие файла для чтения/записи, ввод-вывод в файл и т.д.

Постановка задачи

1. Создать текстовый файл, в котором хранится исходная строка символов, и поместить его в каталог с программой.
2. Составить программу, которая содержит сегменты кода, данных, стека и выполняет:
   - считывание из файла исходной строки.
   - производит над файлом соответствующие варианту действия и выводит результат в другой (новый) созданный файл.
   - при составлении программы использовать цепочечные команды.
   - **Предусмотреть обработку ошибок: файл не найден, ошибка записи, ошибка чтения.**
- Отлаженную программу дополнить комментарием, описывающим ее назначение, и сохранить в файле lab_10.asm для отчета.
3. Написать вариант программы в формате **типа.com**, используя работу со строкой как с массивом символов, запустить ее под управлением отладчика, **проанализировать содержимое регистров процессора**. Отлаженную программу сохранить в файле lab_10_2.asm для отчета.

Содержание отчета:

1. Цель работы.
2. Постановка задачи (текст варианта задания).
3. Теоретическая часть (описание возможных случаев возникновения ошибок и способов их обработки).
4. Листинг программы **типа.exe**.
5. Листинг программы **типа.com**.
6. Анализ содержимого регистров процессора для файла **типа.com**.
7. Пояснения к программе.
8. Вывод.

Варианты:

Вариант 1
Создать новый файл, записав в него из произвольного текстового файла строки длиной менее 4-х символов. Строки показать на экране.

Вариант 2
В четвертой строке произвольного файла заменить заглавные латинские символы на прописные. Строку показать на экране. Изменения сохранить в новом файле.

Вариант 3

Из произвольного текстового файла прочитать 20 последовательных символов, начиная с 10-го байта в файле. Показать строку на экране и записать в новый файл по 4 символа в строке.

Вариант 4
Из произвольного текстового файла прочитать первые 5 символов из каждых последовательных 20 символов. Показать строку на экране и записать, как отдельные строки, в новый файл.

Вариант 5
Прочитать из произвольного текстового файла строки, начинающиеся с символа А, вывести их на экран и записать в новый файл.

Вариант 6
Из второй строке произвольного текстового файла, вывести только слова без повторения букв. Исходную и измененную строку показать на экране. Изменения сохранить в новом файле.

Вариант 7
Во второй строке произвольного текстового файла, у всех слов удалить предыдущие вхождения последней буквы. Исходную и измененную строку показать на экране. Изменения сохранить в новом файле.

Вариант 8
Во второй строке произвольного текстового файла в словах нечетной длины удалить среднюю букву. Исходную и измененную строку показать на экране. Изменения сохранить в новом файле.

Вариант 10
Во второй строке произвольного текстового файла изменить последовательность символов на обратную. Исходную и измененную строку показать на экране. Изменения сохранить в новом файле.

Вариант 11
Из произвольно заданного файла считать первую и последнюю строки, объединить в одну, разбить пополам и записать две строки в новый файл. Все строки показать на экране.

Вариант 12
Найти в произвольном текстовом файле самую длинную строку и самую короткую строку. Записать их в новый файл и показать на экране.

Вариант 13
Заменить первую строку символов в произвольном файле на такое же количество символов, введенных с клавиатуры. Количество символов для ввода запросить после определения длины строки. Исходную и измененную строки показать на экране.

Вариант 14
Из произвольного файла выбрать все цифровые символы и записать последовательно в новый файл, вставляя управляющие коды Возврат каретки и Перевод строки после каждых 10 символов.

Вариант 15
Из каждой строки произвольного текстового файла удалить первые 3 символа и сохранить изменения в новом файле. Исходные и измененные строки показать на экране.

Вариант 16
В начале каждой строки произвольного текстового файла добавить символы, введенные с клавиатуры. Преобразованные строки сохранить в новом файле. Исходные и измененные строки показать на экране.

Вариант 17
Считать из произвольного файла последние 6 символов из каждых 20 последовательных символов. Исходную и измененную строку показать на экране. Записать в новый файл как отдельные строки.

Вариант 18
В произвольном текстовом файле поменять местами первую и последнюю строки. Найденные строки показать на экране. Изменение записать в новый файл.

Вариант 19
Дополнить каждую строку произвольного текстового файла символами "*", по количеству равными номеру строки. Измененные строки показать на экране и записать в новый файл.

Вариант 20
Переписать из произвольного текстового файла в новый файл все строки, заканчивающиеся символом "!" Выбранные строки показать на экране

Теоретическая часть

Программы типа .COM

В некоторых случаях удобно не дробить программу на отдельные сегменты, а включить все компоненты в один сегмент. Этот единственный сегмент должен содержать префикс программы (PSP), коды команд, данные, стек. Такие односегментные программы соответствуют конечной модели типа .com. Программы типа .com не имеют особых преимуществ перед программами .exe, кроме своей компактности, однако широко используется, особенно в качестве резидентных программ. При создании программ типа .com необходимо выполнение двух условий:

1. Исходный текст программы должен быть написан в определенном формате, с ограничениями, соответствующими минимальной модели памяти.
2. Описание всех переменных должно следовать в конце кода программы.
3. После создания объектного модуля, компилятор tlink.exe необходимо запустить с ключом /t для создания программы типа .com.

Пример программы типа .com.

```
sc    segment     'code'
      assume      cs:sc, ds:sc
      org   256         ;резервирование места для PSP
start proc
      mov   AH, 09h
      mov   DX, offset string
      int   21h
      mov   AX, 4C00h
      int   21h
string      db    'Строка для вывода в файле .com'
```

```
start endp
sc    ends
end   start
```

После загрузки программы типа .com регистр IP инициируется числом 256 (100h), поэтому вслед за org 256 должна стоять первая выполняемая строка программы. Образ памяти программы типа .com:

Работа с файлами

Функции DOS int 21h для работы с файлами

Название	***Номер функции***	***Вход***	***Выход***
Создание файла	AH = 3Ch	CX = атрибут файла бит 7: файл можно открывать разными процессорами в Novell Netware бит 6: не используется бит 5: архивный бит (1, если файл не сохранялся) бит 4: директория (должен быть 0 для функции 3Ch) бит 3: метка тома (игнорируется функцией 3Ch) бит 2: системный файл бит 1: скрытый файл бит 0: файл только для чтения DS:DX = адрес ASCIZ - строки с полным именем файла	CF=0 и AX= идентификатор файла, если не произошла ошибка CF=1 и AX = 03h, если путь не найден CF=1 и AX=04h, если слишком много открытых файлов CF=1 и AX = 05h, если доступ запрещен

Открыть существующий файл	AH = 3Dh	AL=режим доступа бит0: открыть для чтения бит1: открыть для записи биты2-3: зарезервированы (0) биты 6-4: режимы доступа для других процессов: 000: режим совместимости (остальные процессы тоже должны открывать этот файл в режиме совместимости) 001: все операции запрещены 010: запись запрещена 011: чтение запрещено 100: запрещений нет бит7: файл не наследуется порождаемыми процессами DS:DX=адрес ASCIZ-строки с полным именем файла CL=маска атрибутов файла	CF=0 и AX= идентификатор файла, если не произошла ошибка CF=1 и AX = код ошибки (02h - файл не найден, 03h - путь не найден, 04h - слишком много открытых файлов, 05h - доступ запрещен, 0Ch - неправильный режим доступа)
Чтение из файла или устройства *(Если при чтении из файла число фактически считанных байтов в AX меньше, чем заказанное в CX, то был достигнут конец файла.)*	AH=3Fh	BX=идентификатор CX=число байтов DS:DX=адрес буфера для приема данных	CF=0 и AX=число считанных байтов, если не было ошибки CF=1 и AX=05h, если доступ запрещен; 06h, если неправильный идентификатор
Переместить указатель чтения/записи	AH=42h	BX=идентификатор CX: DX=расстояние, на которое надо переместить указатель (со знаком) AL = перемещение: 0 - от начала файла 1 - от текущей позиции 2 - от конца файла	CF = 0 и CX:DX = новое значение указателя (в байтах от начала файла), если не произошла ошибка CF = 1 и AX = 06h, если неправильный идентификатор
Запись в файл или устройство	AH=40h	BX=идентификатор CX=число байтов DS:DX=адрес буфера с данными	CF=0 и AX=число считанных байтов, если не произошла ошибка CF=1 и AX=05h, если доступ запрещен; 06h, если неправильный идентификатор

Закрытие файла (Если файл был открыт для записи, все файловые буфера сбрасываются на диск, устанавливается время модификации файла и записывается его новая длина.)	AH=3Eh	BX=идентификатор	CF=0, если не произошла ошибка CF=1 и AX=6, если неправильный идентификатор

Пример считывания из файла sentence.txt строки, запись ее в другой файл newfile.txt и вывод сообщения об успешном выполнении программы:

```
.model small
.stack 100h
.data
;===========================
CR = 0Dh
LF = 0Ah

FileName db "sentence.txt0", "$"          ;имя файла в формате ASCIIZ строки
FDescr dw ?                               ;ячейка для хранения дисриптора

NewFile db "newfile.txt0", "$"
FDescrNew dw ?                            ;для хранения дискриптора нового
файла

Buffer dw ?                               ;буфер для хранения символа строки
String dw 40 dup(0)                       ;буфер для хранения строки
index dw 0                                ;впомогательная переменная

MessageError1 db CR, LF, "File was not opened !", "$"
MessageError2 db CR, LF, "File was not read !", "$"
MessageError3 db CR, LF, "File was not founded!", "$"
MessageError4 db CR, LF, "File was not created!", "$"
MessageError5 db CR, LF, "Error in writing in the file!", "$"

MessageEnd db CR, LF, "Program was successfully finished!", "$"

;===========================

.code

print_string macro
mov ah, 09h
int 21h
endm

start:
      mov ax, @data
      mov ds, ax
```

```
      ;открытие файла
      mov ah, 3Dh
      xor al, al                          ;открыть файл для чтения
      mov dx, offset FileName             ;адрес имени файла
      xor cx, cx                          ;открыть файл без указания атрибутов
      int 21h                             ;выполнить прерывание
      mov FDescr, ax                      ;получить дескриптор файла
      jnc M1                       ;если ошибок нет, выполнить программу дальше
      jmp Er1                             ;файл не был открыт

M1:
    ;создание нового файла
    mov ah, 3ch                           ;создать новый файл
    xor cx, cx
    mov dx, offset NewFile                ;адрес имени файла
    int 21h                               ;выпонить
    mov FDescrNew, ax                     ;дискриптор файла
    jnc M2                         ;если ошибок нет, выполнить программу дальше
    jmp Er3                               ;файл не был создан

M2:
      ;чтение файла
    mov ah, 3fh                           ;чтение из файла
    mov bx, FDescr                        ;дескриптор нужного файла
    mov cx, 1                             ;количество считываемых символов
    mov dx, offset Buffer                 ;адрес буфера для приема
    int 21h                               ;выполнить
    jnc M3                              ;если нет ошибки -> продолжить чтение
    jmp Er2                                 ;если ошибка -> выход
    M3:
        cmp ax, 0 ;если ax=0(число считанных байтов) -> файл кончился -> выход
        je M4                             ;если ax=0 -> sf=1
        mov ax, Buffer
        mov bx, index
        mov String[bx], ax
        inc bx
        mov index, bx
    jmp M2

M4:
    ;запись в файл
    mov ah, 40h
    mov bx, FDescrNew
    mov cx, index
    mov dx, offset String
    int 21h
    jnc M5
    jmp Er4

M5:
    ;закрытие исходного файла
    mov ah, 3eh                           ;функция закрытия файла
    mov bx, FDescr
    int 21h

    ;закрытие нового файла
    mov ah, 3eh                           ;функция закрытия файла
    mov bx, FDescrNew
    int 21h

    ;вывод сообщения об успешном выполнении программы
    mov dx, offset MessageEnd
    print_string
    jmp Exit
```

```
Er1:
    ;файл не был найден
    cmp ax, 02h
    jne M6
    lea dx, MessageError3
    print_string
    jmp Exit
M6:
    ;файл не был открыт
    lea dx, MessageError1
    print_string
    jmp Exit

Er2:
    ;файл не был прочтен
    lea dx, MessageError2
    print_string
    jmp Exit

Er3:
    ;файл не был создан
    lea dx, MessageError4
    print_string
    jmp Exit

Er4:
      ;ошибка при записи в файл
      lea dx, MessageError5
    print_string
    jmp Exit

Exit:
    mov ah, 07h  ;задержка  экрана
    int 21h

    ;завершение программы
    mov ax, 4c00h
    int 21h
end start
```