Міністерство освіти та науки України

# Кіровоградський Державний Технічний університет

Кафедра програмного забезпечення

**Курсовий проект**

з дисципліни

“Програмування на мові ASM-86”

на тему:

“Програма для перегляду великих текстових файлів, розмір яких більший за 64 кілобайти”

**ЗМІСТ**

Завдання

1. Вступ
2. Постановка задачі
3. Обґрунтування вибору методів розв’язку задачі
4. Алгоритм програми
5. Реалізація програми
6. Системні вимоги
7. Інструкція для користувача
8. Висновки.
9. Використана література.

Лістинг програми.

**1. Вступ**

Для перегляду текстових файлів використовуються програми – переглядачі. Вони дозволяють користувачу роздивлятись будь-яку частину файла у зручному для нього вигляді.

Деякі переглядачі мають різні обмеження, наприклад на об’єм файла. Інші можуть переглядати файли будь-якої довжини. Мета цієї роботи – показати програму-переглядач саме для великих файлів.

**2. Постановка задачі**

Необхідно створити програму, з допомогою якої можна було б переглядати текстові файли, за об’ємом більші за 64 кілобайти.

**3. Обґрунтування вибору методів розв’язку задачі**

При перегляді маленьких файлів зручно їх повністю прочитати з диска у пам’ять і виводити на екран звідти. Звичайно, можна застосувати такий спосіб і для великих файлів, але при цьому буде витрачатися багато оперативної пам’яті в машині.

Для того щоб не читати файл повністю, його читають по частинах. Читають якийсь блок і переглядають його. Коли перегляд блоку завершується, то з диска підчитується наступний блок або попередній. Отже, це дає змогу переглядати великі файли при невеликому об’ємі пам’яті.

**4. Алгоритм програми**

1. Ввести ім’я файла

2. Відкрити файл і прочитати перший блок.

3. При натисненні клавіші “вниз” прокрутити текст вниз.

4. При натисненні клавіші “вгору” прокрутити текст вгору.

5. Вивести на екран зміст файлу.

6. Якщо область перегляду вийшла за межі блоку, то прочитати наступний або попередній блок.

7. Якщо користувач не натиснув ESC, перейти до пункту 3.

8. Стоп

**5. Реалізація програми**

Для операцій з файлами використовуються засоби ДОС.

Програма складається з таких процедур:

key\_up - обробка “клавіша вгору”

key\_down - обробка “клавіша вниз”

openfile - відкриття файла

closefile - закриття файла

readnextblock - читання наступного блоку

readprevblock - читання попереднього блоку

initblocks - ініціалізація програми

cleanup - очистка

showframe - відображення кадру

key\_pgup - обробка клавіші Page Up

key\_pgdn - обробка клавіші Page Down

**6. Системні вимоги**

Операційна система МS-DOS

Процесор 8086 або старший

Екран CGA або старший

Пам’ять 640 К

### 7. Інструкція для користувача

Для запуску програми необхідно виконати файл view64.exe. Після нього в командному рядку вводиться ім’я файла. Наприклад, “view64 big\_text.txt”. Після цього на екрані з’явиться зміст файлу. Його можна “гортати” клавішами “Page Up”, “Page Down”, “Up”, “Down”.

Для виходу з програми треба натиснути клавішу “ESC”.

#### 8. Висновки

Отже, на мові Асемблера для процесорів сімейства INTEL x86 створена програма, яка призначена для перегляду в текстовому режимі файлів. Її особливістю є те, що розмір файлів може бути більшим за 64 кілобайти.

Програма працює, але все ж таки може містити помилки. Але робота з нею не вимагає від користувача якихось спеціальних навичок, вона проста в користуванні.

**9. Використана література**

1. Р. Журден. Справ очник программиста на персональном компьютере фирмы IBM.

2. П. Абель. Асемблер и программирование для IBM PC.

**Додаток. ЛІСТИНГ ПРОГРАМИ**

.model compact

.stack 1024

.386

.data

;буфери

pvideobuffer db 4000 dup(0)

fileblock db 32000 dup(0) ;блок з файла

video db 2000 dup(0) ;копiя кадра

errormsg db 'Помилка читання файла!$'

filename0 db 50 dup(0)

msg1 db 'Введiть iм`я файла. $',10,13

filehandle dw 0 ;зсилка

nblock dw 0

currentposbegin dw 0 ;початок блоку

currentposend dw 0 ;кiнець блоку

blocklength dw 0 ;фактична довжина блоку

lastkey db 0 ;останнiй рух: 0 = вниз, 1 - вгору

sym\_count dw 0 ;кiлькiсть символiв на екранi

upperend dw 0

block\_num dw 0

.code

start:

jmp begin

;====================ОБРОБКА КЛАВIШ

;====================KEY\_UP

key\_up proc

pusha

cmp word ptr fs:[lastkey],0

jne ok\_up1

;перевiрка: якщо end<begin то пiдчитати попереднiй блок

mov si,word ptr fs:[currentposend]

mov di,word ptr fs:[currentposbegin]

cmp si,di

ja ok\_up1

cmp word ptr fs:[block\_num],1

jne ok\_up1

;пiдчитка

call readprevblock

mov si,word ptr fs:[blocklength]

add si,offset fileblock

;sub si,word ptr fs:[sym\_count]

mov word ptr fs:[currentposend],si

ok\_up1:

mov word ptr fs:[lastkey],1

;зсовуємо video вниз на length(рядка який вище) байт

;пiдраховуємо length

mov ax,seg video

mov es,ax

mov ds,ax

xor dx,dx ;dx=length[0]

mov bx,word ptr fs:[currentposbegin]

dec bx

get\_len24:

dec bx

cmp byte ptr [bx],10 ;кiнець рядка

je stop\_len24 ;стоп - цикл

good\_sym:

inc dx

cmp dx,79

jb get\_len24

stop\_len24:

inc dx

;тепер dx - довжина 1 рядка

;зсунути video на dx байт вниз

mov di,offset video+2000

mov si,offset video+2000

sub si,dx

std

mov cx,2000+1

rep movsb

;дописати dx байт

;встановлюємо 2000-dx

push ds

mov ax,seg currentposbegin

mov ds,ax

mov si,word ptr ds:[currentposbegin]

mov bx,word ptr ds:[currentposend]

mov di,offset video

add di,dx

pop ds

mov ax,seg fileblock

mov ds,ax

mov cx,dx

;перенос

trans\_u:

dec si

dec di

mov al,byte ptr ds:[si]

mov byte ptr es:[di],al

cmp bx,offset fileblock

jae line\_u\_ok1

mov bx,offset fileblock

add bx,word ptr fs:[blocklength]

line\_u\_ok1:

cmp si,offset fileblock

jae line\_u\_ok

;пiдчитати попереднiй блок

mov word ptr fs:[upperend],1

call readprevblock

mov si,offset fileblock

add si,word ptr fs:[blocklength]

line\_u\_ok:

dec bx

loop trans\_u

stop\_u:

push ds

mov ax,seg currentposend

mov ds,ax

mov word ptr ds:[currentposend],bx

mov word ptr ds:[currentposbegin],si

pop ds

call showframe

popa

ret

key\_up endp

;====================KEY\_DOWN

key\_DOWN proc

pusha

mov word ptr fs:[upperend],0

cmp word ptr fs:[lastkey],1

jne ok\_down1

;перевiрка: якщо end<begin то пiдчитати наступний блок

mov si,word ptr fs:[currentposend]

mov di,word ptr fs:[currentposbegin]

cmp si,di

ja ok\_down1

;пiдчитка

call readnextblock

mov si,offset fileblock

;add si,word ptr fs:[sym\_count]

mov word ptr fs:[currentposbegin],si

ok\_down1:

mov word ptr fs:[lastkey],0

;зсовуємо video вгору на length[0] байт

;пiдраховуємо length[0]

mov ax,seg video

mov es,ax

mov ds,ax

xor dx,dx ;dx=length[0]

mov bx,offset video

get\_len0:

cmp byte ptr [bx],10 ;кiнець рядка

je stop\_len0 ;стоп - цикл

inc bx

inc dx

cmp dx,79

jb get\_len0

stop\_len0:

inc dx

;тепер dx - довжина 1 рядка

;зсунути video на dx байт вгору

mov di,offset video

mov si,offset video

cld

add si,dx

mov cx,1000

rep movsw

;дописати dx байт

;встановлюємо 2000-dx

push ds

mov ax,seg currentposend

mov ds,ax

mov si,word ptr ds:[currentposend]

mov bx,word ptr ds:[currentposbegin]

mov di,2000+offset video

sub di,dx

pop ds

mov ax,seg fileblock

mov ds,ax

mov cx,dx

;перенос

trans\_d:

mov al,byte ptr ds:[si]

mov byte ptr es:[di],al

inc si

inc di

inc bx

mov bp,offset fileblock

add bp,word ptr fs:[blocklength]

cmp bx,bp

jbe line\_d\_ok1

mov bx,offset fileblock

line\_d\_ok1:

cmp si,bp

jbe line\_d\_ok

;пiдчитати наступний блок

call readnextblock

mov si,offset fileblock

;EOF

line\_d\_ok:

loop trans\_d

stop\_d:

push ds

mov ax,seg currentposend

mov ds,ax

mov word ptr ds:[currentposend],si

mov word ptr ds:[currentposbegin],bx

pop ds

call showframe

popa

ret

key\_DOWN endp

;===================вiдкриття файла

openfile proc

pusha

mov ah,3dh

xor al,al

mov dx,seg filename0

mov ds,dx

mov dx,offset filename0

int 21h

jc error

mov word ptr ds:[filehandle],ax

popa

ret

error:

call cleanup

mov ah,9

mov dx,seg errormsg

mov ds,dx

mov dx,offset errormsg

int 21h

popa

mov ah,4ch

int 21h

openfile endp

;===================закриття файла

closefile proc

pusha

mov ah,3eh

mov dx,seg filehandle

mov ds,dx

mov bx,word ptr ds:[filehandle]

int 21h

popa

ret

closefile endp

;===================читання наступного блоку з файла

readnextblock proc

pusha

mov ax,seg filehandle

mov ds,ax

mov ax,word ptr fs:[block\_num]

inc ax

mov word ptr fs:[block\_num],ax

retry\_n:

mov ah,3fh

mov bx,word ptr ds:[filehandle]

mov cx,32000

mov dx,seg fileblock

mov ds,dx

mov dx,offset fileblock

int 21h

;EOF - заповнити 0

cmp ax,0

jne read\_ne

mov ah,42h

mov al,0

mov bx,word ptr ds:[filehandle]

xor cx,cx

xor dx,dx

int 21h ;==> на початок

jmp retry\_n

read\_ne:

mov dx,seg blocklength

mov ds,dx

mov word ptr ds:[blocklength],ax ;довжина блоку

popa

ret

readnextblock endp

;===================читання попереднього блоку з файла

readprevblock proc

pusha

mov ax,seg filehandle

mov ds,ax

mov ax,word ptr fs:[block\_num]

dec ax

mov word ptr fs:[block\_num],ax

;seek - спочатку поточне змiщення

mov ah,42h

mov al,1

mov bx,word ptr ds:[filehandle]

xor cx,cx

xor dx,dx

int 21h ;==> DX:AX - поточне

;перевiрити, чи є нуль - початок файла. Тодi - змiстити

;на кiнець

cmp dx,0

jne norm\_read

cmp ax,32000

jbe make\_lastpos

jmp norm\_read

;на кiнець

make\_lastpos:

mov ah,42h

mov al,2

mov bx,word ptr ds:[filehandle]

xor cx,cx

xor dx,dx

int 21h ;==> DX:AX - поточне

jmp setpos

norm\_read:

setpos:

sub ax,64000

sbb dx,0

mov cx,dx

mov dx,ax

mov ah,42h

mov al,0

mov bx,word ptr ds:[filehandle]

int 21h ;==> змiстили

mov ah,3fh

mov bx,word ptr ds:[filehandle]

mov cx,32000

mov dx,seg fileblock

mov ds,dx

mov dx,offset fileblock

int 21h

mov dx,seg blocklength

mov ds,dx

mov word ptr ds:[blocklength],ax ;довжина блоку

popa

ret

readprevblock endp

;======================== iнiцiалiзацiя блокiв

initblocks proc

pusha

call readnextblock

;1 копiя в video

mov dx,seg video

mov es,dx

mov di,offset video

mov dx,seg fileblock

mov ds,dx

mov si,offset fileblock

cld

mov cx,1000

rep movsw

;початковi значення кордонiв - 0 i 2000

mov ax,offset fileblock

mov word ptr ds:[currentposbegin],ax

mov bx,2000+offset fileblock

mov word ptr ds:[currentposend],bx

popa

ret

initblocks endp

;------------------------ процедура очистки

cleanup proc

;очистка екрана

cld

push es

mov cx,80\*25

mov ax,0b800h

mov es,ax

xor di,di

mov ax,0700h

rep stosw

pop es

;поставити нормальний курсор

mov ah,1

xor cl,cl

mov ch,30h

int 10h

ret

cleanup endp

;---------------------- процедура вiдображення кадра

showframe proc

pusha

mov ax,seg pvideobuffer

mov ds,ax

;настройка кольору

;верхнiй рядок сiрий

mov bx,offset pvideobuffer

mov cx,80

setw: mov byte ptr ds:[bx],32

mov byte ptr ds:[bx+1],70h

add bx,2

loop setw

;синiй фон, жовтий текст

mov cx,80\*24

setm: mov byte ptr ds:[bx],32

mov byte ptr ds:[bx+1],1eh

add bx,2

loop setm

;кольори

;??????????????????=================================

;виведення video

;сегменти i вказiвники

;si - номер символа в буферi

;di - iндекс байта вiдеопам@ятi

;виводимо всi символи як є, але!

;13-й пропускаємо, 10-й - перехiд на

;новий рядок, а 9 - табулюємо (10)

mov ax,seg video

mov ds,ax

mov ax,seg pvideobuffer

mov es,ax

mov si,offset video

mov di,offset pvideobuffer

;пропустити 1 рядок

add di,80\*2

symb:

mov al,byte ptr ds:[si] ;символ

cmp al,10 ;перехiд на новий рядок

jne normal

;почати новий рядок

push ax

push bx

push dx

push di

add di,160 ;di=di+ (160-((di+160) mod 160))

mov ax,di

xor dx,dx

mov bx,160

div bx

mov ax,160

sub ax,dx

pop di

add di,ax

pop dx

pop bx

pop ax

jmp skip; новий рядок

normal:

cmp al,13

je skip

;нормальний символ - виводимо

mov byte ptr es:[di],al

add di,2 ;2 байта - наступнi (атрибут пропустити)

skip:

inc si ;наступний байт iнформацiї

push si

sub si,offset video

mov word ptr fs:[sym\_count],si

pop si

cmp di,4000 ;кiнець екрана

jb symb

;??????????????????===================================

;на екран виводимо з пам@ятi

cld

push es

mov cx,80\*25

mov ax,0b800h

mov es,ax

xor si,si

xor di,di

rep movsw

pop es

popa

ret

showframe endp

;====================KEY\_PGUP

key\_pgup proc

;24 рази викликати key\_up

push cx

mov cx,24

scr\_up:

call key\_up

loop scr\_up

pop cx

ret

key\_pgup endp

;====================KEY\_PGDN

key\_pgdn proc

;24 рази викликати key\_down

push cx

mov cx,24

scr\_dn:

call key\_down

loop scr\_dn

pop cx

ret

key\_pgdn endp

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

begin:

mov ax,seg msg1

mov ds,ax

;введення iменi файла 1

mov ah,9

mov dx,offset msg1

int 21h

mov bx,offset filename0

ent\_1:

mov ah,1h

int 21h

mov [bx],al

inc bx

cmp al,13

jne ent\_1

dec bx

mov byte ptr [bx],0

mov ah,2

mov dl,10

int 21h

mov ax,seg blocklength

mov fs,ax

;сховати курсор

mov ah,1

xor cl,cl

mov ch,00000010b

int 10h

call openfile

call initblocks

call showframe

;чекаємо клавiшу

wait\_key:

xor ah,ah

int 16h

;al-ascii code

cmp ah,1 ;ESC

je halt

cmp ah,48h ;UP

jne next1

call key\_up

jmp wait\_key

next1: cmp ah,50h

jne next2 ;DOWN

call key\_down

jmp wait\_key

next2: cmp ah,49h ;PGUP

jne next3

call key\_pgup

jmp wait\_key

next3: cmp ah,51h ;PGDN

jne wait\_key

call key\_pgdn

jmp wait\_key

halt:

call cleanup

call closefile

mov ah,4ch

int 21h

end start