Міністерство освіти та науки України

# Кіровоградський Державний Технічний університет

Кафедра програмного забезпечення

**Курсовий проект**

з дисципліни

«Програмування на мові ASM-86»

на тему:

«Побудова дерева каталогів диску і реалізація можливості переходу у вибраний каталог»

**Зміст**

Завдання

Вступ

1. Постановка задачі
2. Обґрунтування вибору методів розв’язку задачі
3. Алгоритм програми
4. Реалізація програми
5. Системні вимоги
6. Інструкція для користувача

Висновки

Використана література

Лістинг програми

**Вступ**

Як відомо, дискова інформація в комп’ютерах записується у вигляді файлів. На дисках, крім файлів, існують також директорії. Завдяки цьому на диску можна групувати файли.

Структура інформації на диску нагадує дерево. Так, кореневий каталог – це «вершина» дерева, а нижче ідуть «гілки» піддиректорій.

Для перегляду дерева каталогів і «мандрів» по ньому можна використовувати засоби ДОС або програми-оболонки. Можна також створити спеціальні програми. У даній роботі розглянута програма побудови дерева каталогів диску, вона дає змогу вибрати і перейти в потрібний каталог.

1**. Постановка задачі**

Використовуючи мову асемблера, написати програму, яка б дозволяла відобразити на екрані дерево каталогів на диску і перейти в потрібний користувачеві каталог.

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

Для пошуку каталогів на диску доцільно використовувати функції DOS переривання 21. Перша з них (4еh) шукає перший файл, який підходить до шаблону і атрибутів, і 4fh, яка шукає наступний файл. При знайденні каталогу відбувається перехід в нього і знову викликається (рекурсивно) функція аналізу каталога. Таким чином, у пам’яті створюється структура, яка зберігає імена директорій та рівень їх вкладеності.

Після сканування відбувається відображення першого кадру дерева. Керування клавіатурою здійснюється за допомогою переривання 16.

Виведення на екран здійснюється також з використанням функцій ДОС 21 переривання.

**3. Алгоритм програми**

1. Виділити пам’ять

2. Перейти в кореневий каталог

3. Знайти каталог на диску.

4. Перейти в нього і зберегти в пам’яті його ім’я

5. Рекурентно перейти на пункт 3.

6. Якщо каталог не знайдено, то припинити сканування.

7. Очистити екран.

8. Вивести на екран 24 рядки каталогів

9. При натисканні клавіш «вгору», «вниз» прокручувати дерево.

10. При натисненні «ENTER» сформувати шлях і перейти в заданий каталог.

11. Кінець

**4. Реалізація програми**

Після початку роботи програми виділяється пам’ять для зберігання дерева каталогів. Потім за наведеним алгоритмом відбувається рекурсивне сканування дерева каталогів. Для реалізації рекурсії використовується зменшення показника стеку і копіювання області DTA у стек. Після виходу з рекурсії – область DTA для конкретного каталогу відновлюється.

Після сканування екран очищується шляхом запису нулів у відеопам'ять. Після цього засобами 21 переривання у циклі з відступами виводяться назви каталогів в тому порядку, в якому вони записані в пам’яті. Виведення виводить процедура, починаючи з певної позиції. При натисканні клавіш відбувається зміна позиції і перемалювання екрану. Для переходу в заданий каталог використовується функція 3bh 21 переривання.

**5. Системні вимоги**

Програма працює на IBM-PC-сумісних комп’ютерах з мікропроцесором INTEL 8086 або старшим. Необхідно 640 КБайт пам’яті, монітор CGA або старший, а також накопичувач на жорстких чи гнучких магнітних дисках. Необхідна також операційна система MS-DOS або сумісна з нею (Windows).

### 6. Інструкція для користувача

Для роботи треба запустити програму tree.com. Після цього на екрані після деякої паузи (відбувається сканування дерева) виводиться дерево каталогів. По ньому можна переміщуватися клавішами «вгору» та «вниз». Верхній каталог виділено кольором. Якщо натиснути Enter, відбудеться перехід в заданий каталог. Вихід з програми здійснюється за допомогою клавіші ESC.

#### Висновки

Отже, створена програма, яка дозволяє працювати з деревом каталогів диску. Вона є прикладом для роботи на асемблері з функціями пошуку файлів в каталозі, а також роботи з пам’яттю.

**Використана література**

1. Роберт Журден. Справочник программиста на персональном компьютере фирмы IBM

2. П. Абель. Ассемблер для IBM PC

**Лістинг програми**

model tiny

386

code

org 100h

start:

jmp begin

mem\_er db 'memory fault! $'

scan\_msg db 'Scanning…$'

all\_dir db '\*.\*', 0

up\_dir db '..', 0

r\_dir db '\', 0

recurse\_level db 0

names\_offset dw 0

dirnum dw 0

dirpos dw 0

path db '\', 100 dup(0); path

change\_dir proc

mov si, word ptr [dirpos]

mov al, byte ptr fs: [si]; level

inc al

xor ah, ah; counter

mov di, offset path+1

next\_level:

cmp ah, al

je stop\_path

; шукаємо каталог позаду з рiвнем ah

push si

find\_level:

cmp byte ptr fs: [si], ah

je found\_level

sub si, 14

jmp find\_level

found\_level:

inc si

mov dl, byte ptr fs: [si]

cmp dl, 0

je stop\_dir

mov byte ptr [di], dl

inc di

jmp found\_level

stop\_dir:

mov byte ptr [di], '\'

inc di

pop si

inc ah

jmp next\_level

stop\_path:

mov byte ptr [di-1], 0

; chdir

mov ah, 3bh

mov dx, offset path

int 21h

ret

change\_dir endp

scan\_dir proc

mov ah, 2fh

int 21h

; findfirst

mov ah, 4eh

mov cx, 10h

mov dx, offset all\_dir

int 21h

findnext:

jc stop\_find

cmp byte ptr [bx+15h], 10h

jne skip\_nodir

cmp byte ptr [bx+1eh], '.'

je skip\_nodir

mov di, word ptr cs: [names\_offset]

mov al, byte ptr cs: [recurse\_level]

mov byte ptr fs: [di], al

inc di

mov si, bx

add si, 1eh

cld

mov cx, 13

push es

push fs

pop es

rep movsb

pop es

; chdir

mov ah, 3bh

mov dx, bx

add dx, 1eh

int 21h

jc skip\_nodir; error

add word ptr cs: [names\_offset], 14

add word ptr cs: [recurse\_level], 1

add word ptr cs: [dirnum], 1

; save dta

sub sp, 2bh

mov di, sp

mov si, bx

mov cx, 2bh

rep movsb

call scan\_dir

mov si, sp

mov di, bx

mov cx, 2bh

rep movsb

add sp, 2bh

; restore dta

sub word ptr cs: [recurse\_level], 1

; chdir.

mov ah, 3bh

mov dx, offset up\_dir

int 21h

skip\_nodir:

mov ah, 4fh

int 21h

jmp findnext

stop\_find:

ret

scan\_dir endp

show\_dirs proc

; clrscr

draw\_screen:

push es

mov ax, 0b800h

mov es, ax

xor di, di

mov ax, 1c00h

mov cx, 80

rep stosw

mov di, 80

mov cx, 2000–80

mov ax, 0700h

rep stosw

pop es

mov ah, 2

xor dx, dx

xor bh, bh

int 10h

mov cx, 24

mov di, word ptr cs: [dirpos]

draw\_name:

mov si, di

push cx

xor cx, cx

mov cl, byte ptr fs: [si]

inc cx

mov ah, 2

mov dl, ' '

set\_spac:

int 21h

int 21h

int 21h

loop set\_spac

pop cx

inc si

next\_let:

mov dl, byte ptr fs: [si]

cmp dl, 0

je next\_name

mov ah, 2

int 21h

inc si

jmp next\_let

next\_name:

mov ah, 2

mov dl, 10

int 21h

mov dl, 13

int 21h

add di, 14

loop draw\_name

xor ah, ah

int 16h

cmp ah, 50h

jne no\_50h

; down

mov ax, word ptr cs: [dirpos]

add ax, 14

mov word ptr cs: [dirpos], ax

jmp draw\_screen

no\_50h:

cmp ah, 48h

jne no\_48h

; up

mov ax, word ptr cs: [dirpos]

cmp ax, 0

je draw\_screen

sub ax, 14

mov word ptr cs: [dirpos], ax

jmp draw\_screen

no\_48h:

cmp ah, 1

je stop\_s

cmp ah, 1ch

jne draw\_screen

; chdir

call change\_dir

jmp stop\_s

jmp draw\_screen

stop\_s:

ret

show\_dirs endp

begin:

mov bx, 4096

mov ah, 4ah

int 21h

mov ax, bx

shl ax, 4

mov sp, ax

mov ah, 48h

mov bx, 1000h

int 21h

jc mem\_errr

mov fs, ax

push es

push fs

pop es

xor di, di

mov cx, 0ffffh

xor ax, ax

rep stosb

pop es

mov dx, offset scan\_msg

mov ah, 9

int 21h

mov ah, 3bh

mov dx, offset r\_dir

int 21h

call scan\_dir

call show\_dirs

push fs

pop es

mov ah, 49h

int 21h

mov ah, 4ch

int 21h

mem\_errr:

mov ah, 9

mov dx, offset mem\_Er

int 21h

en db?

end start