Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

Кафедра ПЗ

Лабораторна робота №1

з дисципліни «Системне програмування»

Виконав: ст. 2ПІ-22Б Чорний О. В.

Перевірив: доцент Майданюк В. П.

Вінниця – 2025

**ЛАБОРАТОРНА РОБОТА №1**

**КЕРУВАННЯ КЛАВІАТУРОЮ ТА ДИСПЛЕЕМ В ТЕКСТОВОМУ**

**РЕЖИМІ МОВОЮ АСЕМБЛЕР З ВИКОРИСТАННЯМ**

**СЕРЕДОВИЩА TASM**

**Мета**: вивчити особливостi виведення інформації на дисплей

**Завдання 1**: 11. Вивести на екран у текстовому режимі таблицю у вигляді: A 01

B 03

C 00

D 01

..

Z 02

При натисканні відповідної клавіші-букви відповідне число зростає. Вихід з програми - при натискуванні клавіші ESC. Використовувати тільки функції BIOS.

**Код програми 1:**

.model small

.stack 100h

.data

my\_table db 'A 00', 0Dh, 0Ah

db 'B 00', 0Dh, 0Ah

db 'C 00', 0Dh, 0Ah

db 'D 00', 0Dh, 0Ah

db 'E 00', 0Dh, 0Ah

db 'F 00', 0Dh, 0Ah

db 'G 00', 0Dh, 0Ah

db 'H 00', 0Dh, 0Ah

db 'I 00', 0Dh, 0Ah

db 'J 00', 0Dh, 0Ah

db 'K 00', 0Dh, 0Ah

db 'L 00', 0Dh, 0Ah

db 'M 00', 0Dh, 0Ah

db 'N 00', 0Dh, 0Ah

db 'O 00', 0Dh, 0Ah

db 'P 00', 0Dh, 0Ah

db 'Q 00', 0Dh, 0Ah

db 'R 00', 0Dh, 0Ah

db 'S 00', 0Dh, 0Ah

db 'T 00', 0Dh, 0Ah

db 'U 00', 0Dh, 0Ah

db 'V 00', 0Dh, 0Ah

db 'W 00', 0Dh, 0Ah

db 'X 00', 0Dh, 0Ah

db 'Y 00', 0Dh, 0Ah

db 'Z 00', 0Dh, 0Ah, '$'

.code

start:

mov ax, @data

mov ds, ax

mov ah, 0

mov al, 3

int 10h

main\_loop:

mov ah, 02h

mov bh, 0

mov dh, 0

mov dl, 0

int 10h

mov ah, 09h

lea dx, my\_table

int 21h

call clear\_buffer

mov ah, 0

int 16h

push ax

mov ah, 0Eh

mov bh, 0

int 10h

mov al, 0Dh

int 10h

mov al, 0Ah

int 10h

pop ax

cmp al, 1Bh

je exit

cmp al, 'a'

jb check\_upper

cmp al, 'z'

ja check\_upper

sub al, 32

jmp process\_key

check\_upper:

cmp al, 'A'

jb main\_loop

cmp al, 'Z'

ja main\_loop

process\_key:

sub al, 'A'

mov bl, 5

mul bl

add ax, 2

mov si, ax

push ax

push bx

mov ax, si

call print\_number

pop bx

pop ax

mov bl, my\_table[si]

mov bh, my\_table[si+1]

cmp bh, '9'

jne inc\_second

cmp bl, '9'

jne inc\_first

jmp main\_loop

inc\_second:

inc bh

mov my\_table[si+1], bh

jmp clear\_screen

inc\_first:

inc bl

mov my\_table[si], bl

mov my\_table[si+1], '0'

jmp clear\_screen

clear\_screen:

mov ah, 0

mov al, 3

int 10h

jmp main\_loop

exit:

mov ah, 4Ch

int 21h

clear\_buffer:

mov ah, 1

int 16h

jz buffer\_cleared

mov ah, 0

int 16h

jmp clear\_buffer

buffer\_cleared:

ret

print\_number:

push ax

push bx

push cx

push dx

mov bx, 10

xor cx, cx

convert\_loop:

xor dx, dx

div bx

push dx

inc cx

test ax, ax

jnz convert\_loop

print\_loop:

pop dx

add dl, '0'

mov ah, 0Eh

mov al, dl

mov bh, 0

int 10h

loop print\_loop

mov al, 0Dh

int 10h

mov al, 0Ah

int 10h

pop dx

pop cx

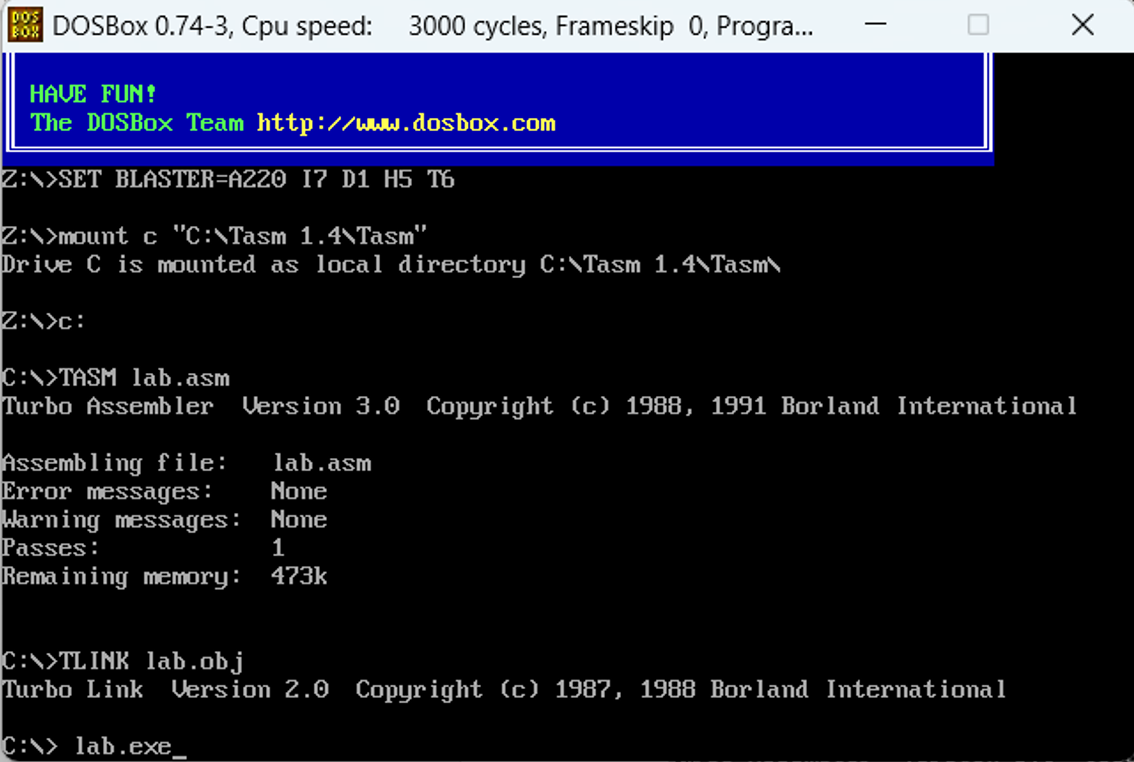
pop bx

pop ax

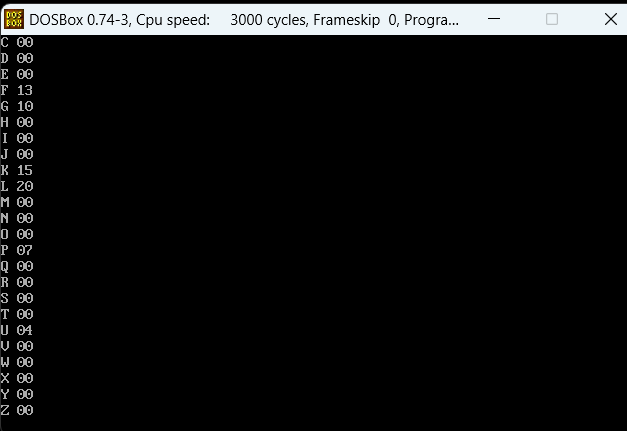
ret

end start

**Запуск програми за допомогою DOSBox:**



**Результат роботи програми:**



**Завдання 2:** 5. Побудувати прямокутник з розмірами сторін 50х100 і повільно повертати його по часовій стрілці на кути кратні 90 градусів.

**Код програми 2:**

.model small

.stack 100h

.data

x1 dw 110

x2 dw 210

y1 dw 50

y2 dw 100

angle db 0

.code

main proc

mov ax, 0013h

int 10h

draw\_loop:

call draw\_rectangle

call delay

call erase\_rectangle

call rotate\_rectangle

jmp draw\_loop

main endp

draw\_rectangle proc

push ax

push bx

push cx

push dx

mov al, 0Fh

mov ah, 0Ch

mov cx, x1

mov dx, y1

top\_line:

int 10h

inc cx

cmp cx, x2

jle top\_line

mov cx, x1

mov dx, y2

bottom\_line:

int 10h

inc cx

cmp cx, x2

jle bottom\_line

mov cx, x1

mov dx, y1

left\_line:

int 10h

inc dx

cmp dx, y2

jle left\_line

mov cx, x2

mov dx, y1

right\_line:

int 10h

inc dx

cmp dx, y2

jle right\_line

pop dx

pop cx

pop bx

pop ax

ret

draw\_rectangle endp

erase\_rectangle proc

push ax

push bx

push cx

push dx

mov al, 0

mov ah, 0Ch

mov cx, x1

mov dx, y1

erase\_top:

int 10h

inc cx

cmp cx, x2

jle erase\_top

mov cx, x1

mov dx, y2

erase\_bottom:

int 10h

inc cx

cmp cx, x2

jle erase\_bottom

mov cx, x1

mov dx, y1

erase\_left:

int 10h

inc dx

cmp dx, y2

jle erase\_left

mov cx, x2

mov dx, y1

erase\_right:

int 10h

inc dx

cmp dx, y2

jle erase\_right

pop dx

pop cx

pop bx

pop ax

ret

erase\_rectangle endp

rotate\_rectangle proc

inc angle

cmp angle, 4

jl no\_reset

mov angle, 0

no\_reset:

cmp angle, 0

je angle\_0

cmp angle, 1

je angle\_90

cmp angle, 2

je angle\_180

cmp angle, 3

je angle\_270

angle\_0:

mov x1, 110

mov x2, 210

mov y1, 50

mov y2, 100

jmp end\_rotate

angle\_90:

mov x1, 85

mov x2, 135

mov y1, 75

mov y2, 175

jmp end\_rotate

angle\_180:

mov x1, 110

mov x2, 210

mov y1, 100

mov y2, 150

jmp end\_rotate

angle\_270:

mov x1, 185

mov x2, 235

mov y1, 25

mov y2, 125

end\_rotate:

ret

rotate\_rectangle endp

delay proc

push cx

push dx

mov cx, 0Fh

mov dx, 4240h

mov ah, 86h

int 15h

pop dx

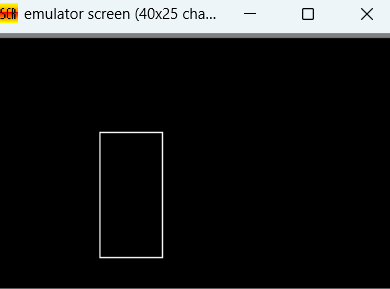
pop cx

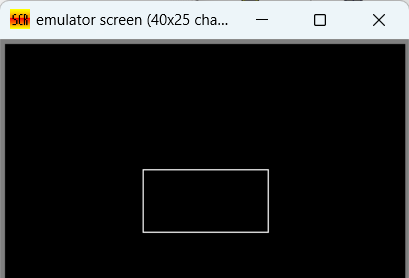
ret

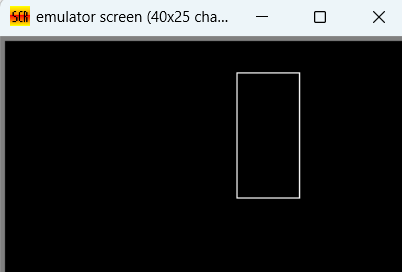
delay endp

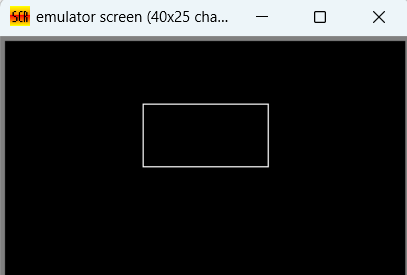
end main

**Реузльтат роботи програми 2:**









**Контрольні питання :**

1. Як встановити графічні режими роботи відеоадаптера?

Графічні режими встановлюються за допомогою виклику BIOS-переривання, яке дозволяє відеоадаптеру перемкнутися в один із доступних графічних режимів, наприклад VGA. Кожен режим має свій номер і визначає роздільну здатність та кількість кольорів.

2. Як записати точку в графічному режимі?

Для запису точки необхідно обчислити її позицію в пам’яті відеоадаптера та встановити відповідне значення, що відповідає кольору. Пам’ять організована відповідно до режиму, тож розташування точки залежить від координат X і Y.

3. Як прочитати точку в графічному режимі?

Щоб прочитати точку, потрібно визначити її положення у відеопам’яті на основі координат і зчитати значення, яке відповідає кольору пікселя в цій точці.

4. Як встановити палітру?

Палітра встановлюється шляхом зміни значень кольорів у таблиці палітри, яка визначає, які реальні кольори відповідають певним номерам кольорів. Це робиться через BIOS або безпосередньо через регістри відеоконтролера.

5. Які графічні режими підтримує стандарт VGA?

Стандарт VGA підтримує кілька графічних режимів, серед яких найбільш відомі: 640x480 з 16 кольорами, 320x200 з 256 кольорами, а також інші режими з меншою або більшою роздільною здатністю та кількістю кольорів.

6. Наведіть алгоритм побудови прямої лінії.

Для побудови прямої лінії між двома точками використовується алгоритм, що поступово обчислює координати пікселів уздовж лінії. Найвідомішим є алгоритм Брезенхема, який дозволяє швидко визначити, які пікселі потрібно зафарбувати, щоб утворити пряму.

7. Як задати колір фону в графічному режимі?

Колір фону задається або при очищенні екрану шляхом заповнення всієї відеопам’яті значенням певного кольору, або за допомогою зміни системного кольору фону в регістрах відеоадаптера (для текстових режимів це окреме поле в атрибуті символу).