**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №6**

**по дисциплине «Операционные системы»**

Тема: **Построение модуля динамической структуры**

|  |  |  |
| --- | --- | --- |
| Студент гр. 6383 |  | Медведев Г.О. |
| Преподаватель |  | Губкин А.Ф. |

Санкт-Петербург

2018

**Цель работы.**

Исследование возможности построения загрузочного модуля динамической структуры.

**Описание функций и структур данных.**

|  |  |
| --- | --- |
| **Название функции** | **Назначение** |
| PRINT | вызывает функцию печати строки |
| BYTE\_TO\_HEX | переводит число AL в коды символов 16-ой с/с, записывая получившееся в al и ah |
| TETR\_TO\_HEX | вспомогательная функция для работы функции BYTE\_TO\_HEX |
| DEAL | запускает вызываемый модуль |
| PAR\_BL | создаёт блок параметров |
| MEMORY\_ | освобождает лишнюю память |

**Последовательность действий, выполняемых утилитой.**

1. Освобождает память и обрабатывает ошибки, которые могли произойти.
2. Создаёт блок параметров.
3. Запускает вызываемый модуль.
4. Выводит код завершения при отсутствии ошибок, при их наличии – обрабатывает их.

**Результаты выполнения программ.**

1. Запуск программы 6.asm, вызывающей программу 2.com, которая останавливается, ожидается ввод символа. Действие изображено на Рис. 1.

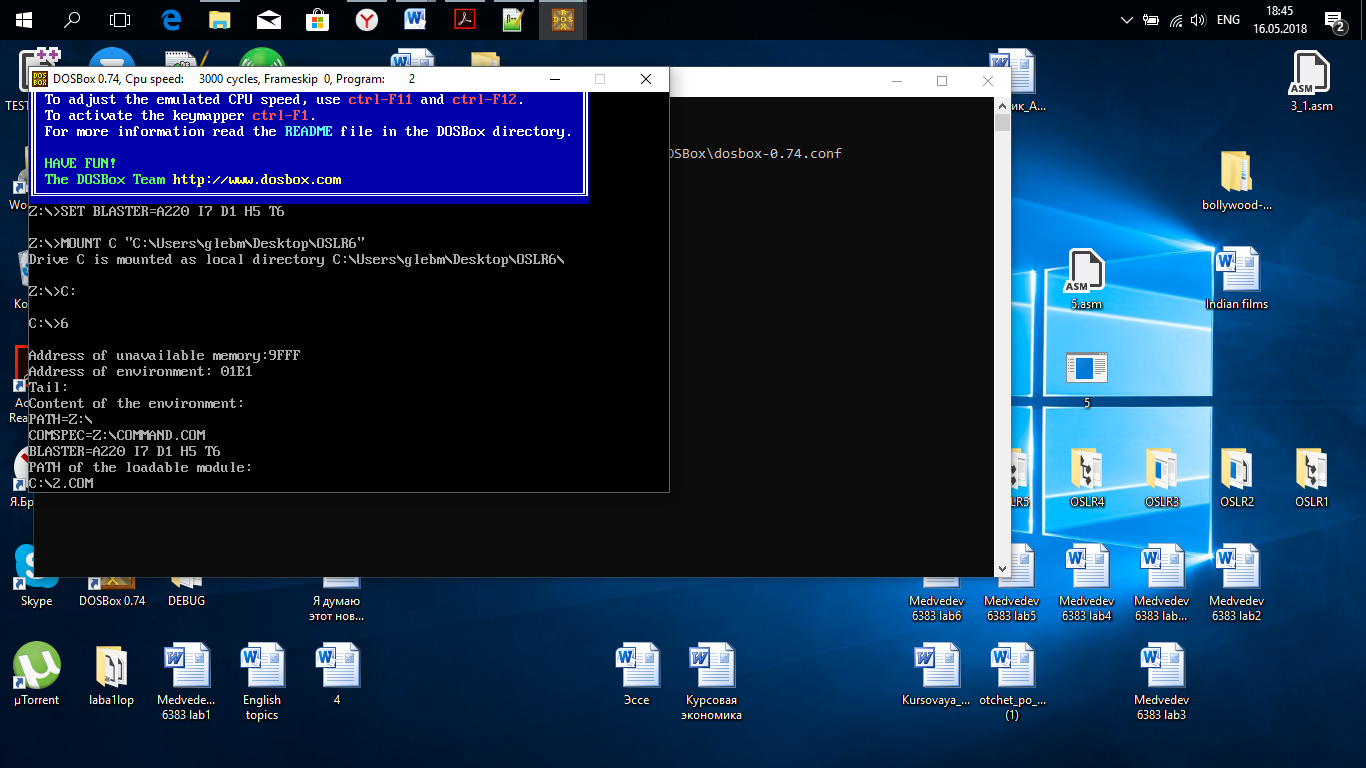


Рисунок 1 – Результат выполнения программы 6.exe

1. На Рис. 2 представлен ввод символа.

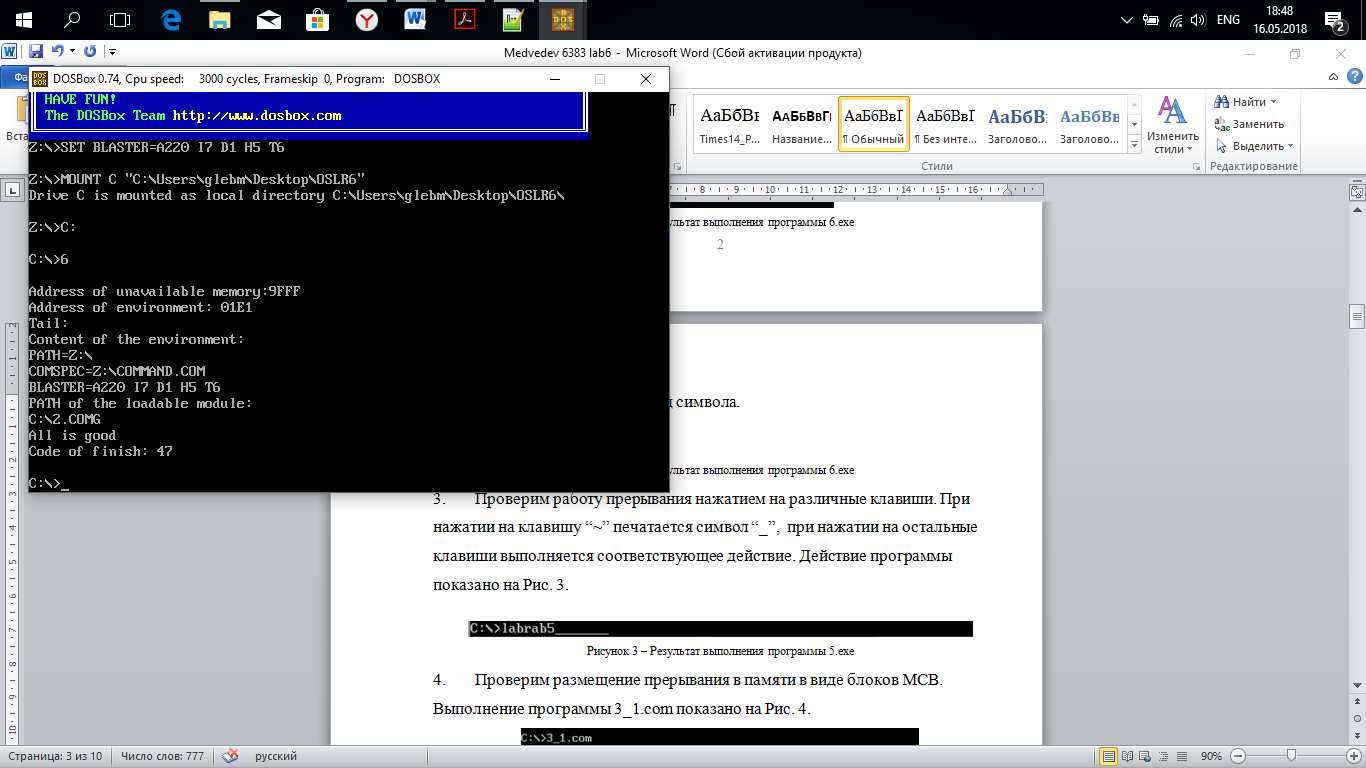


Рисунок 2 – Результат выполнения программы 6.exe

1. Запустим программу снова, теперь введём сочетание клавиш Ctrl+C. Действие программы показано на Рис. 3.

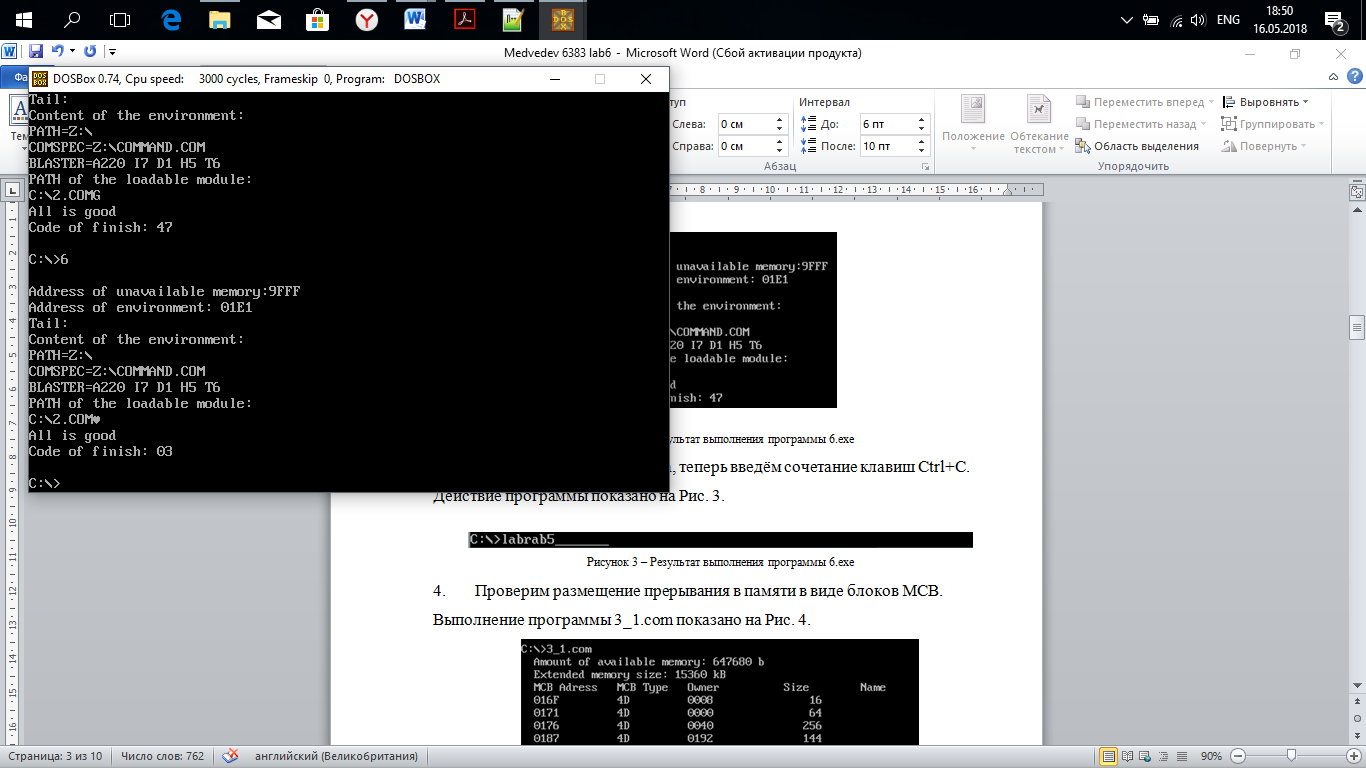


Рисунок 3 – Результат выполнения программы 6.exe

1. Запустим программу 6.exe, когда оба файла не находятся в текущем каталоге. Выполнение программы 6.exe показано на Рис. 4.

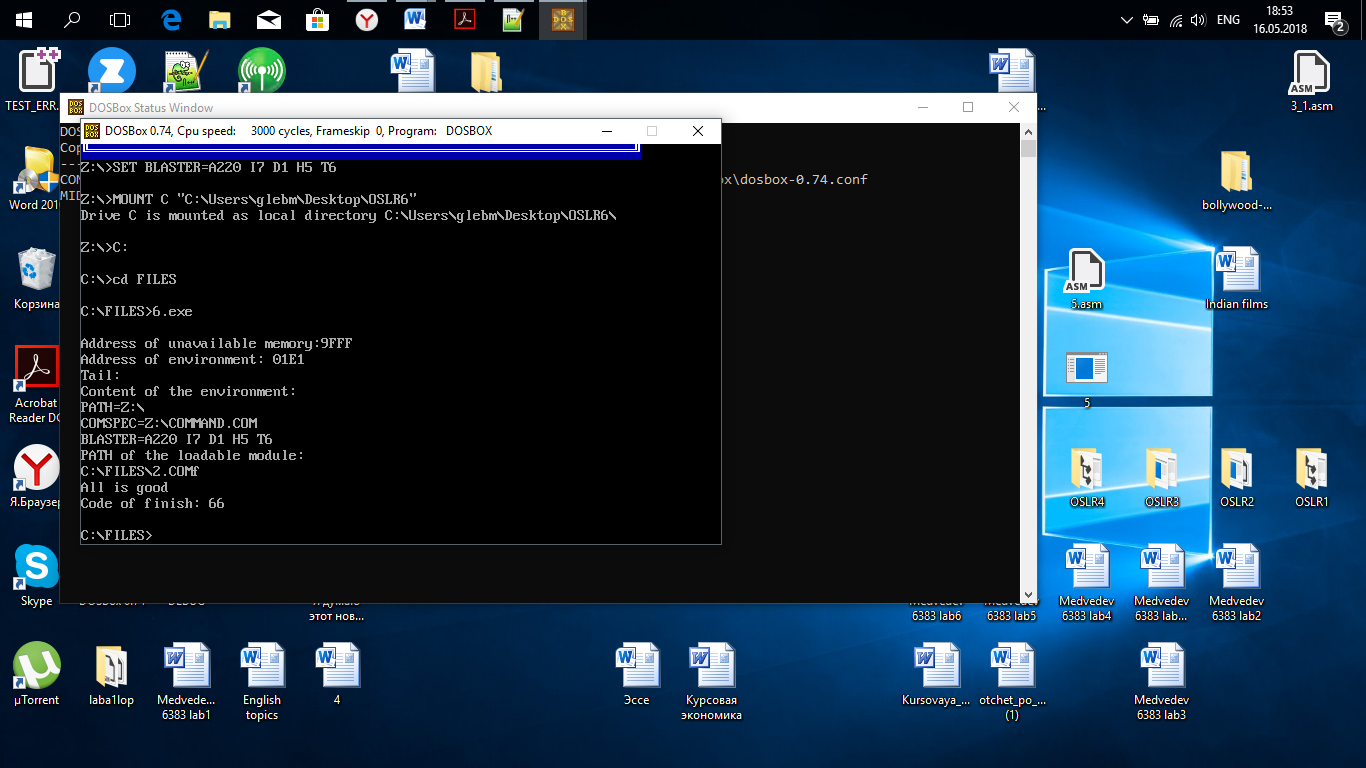


Рисунок 4 – Результат выполнения программы 6.exe

1. Запустим программу снова, теперь введём сочетание клавиш Ctrl+C. Действие программы показано на Рис. 5.

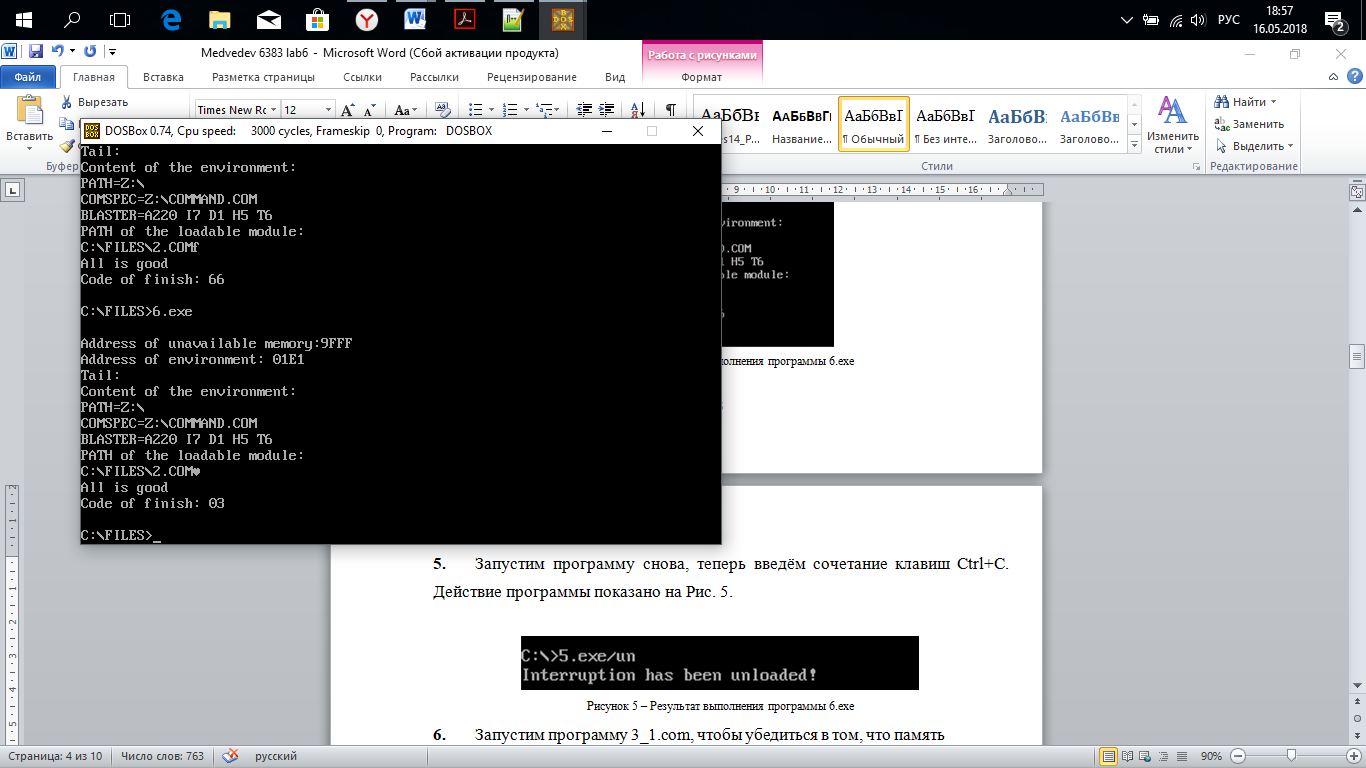


Рисунок 5 – Результат выполнения программы 6.exe

1. Запуск программы, когда оба файла в разных каталогах, при этом указан путь до второго файла. Выполнение изображено на Рис. 6.

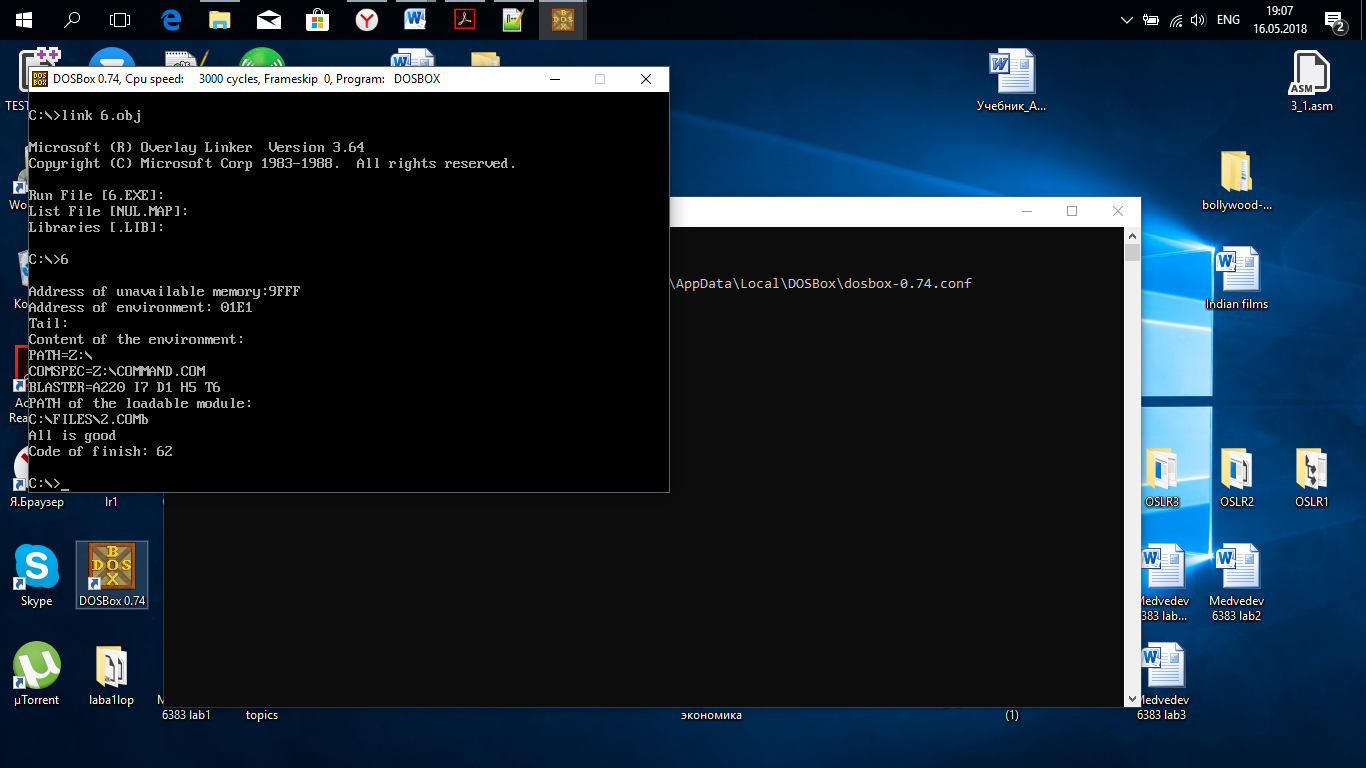


Рисунок 6 – Результат повторного выполнения программы 6.exe

1. Запуск программы, когда файлы находятся в разных каталогах.Результат выполнения на Рис. 7.

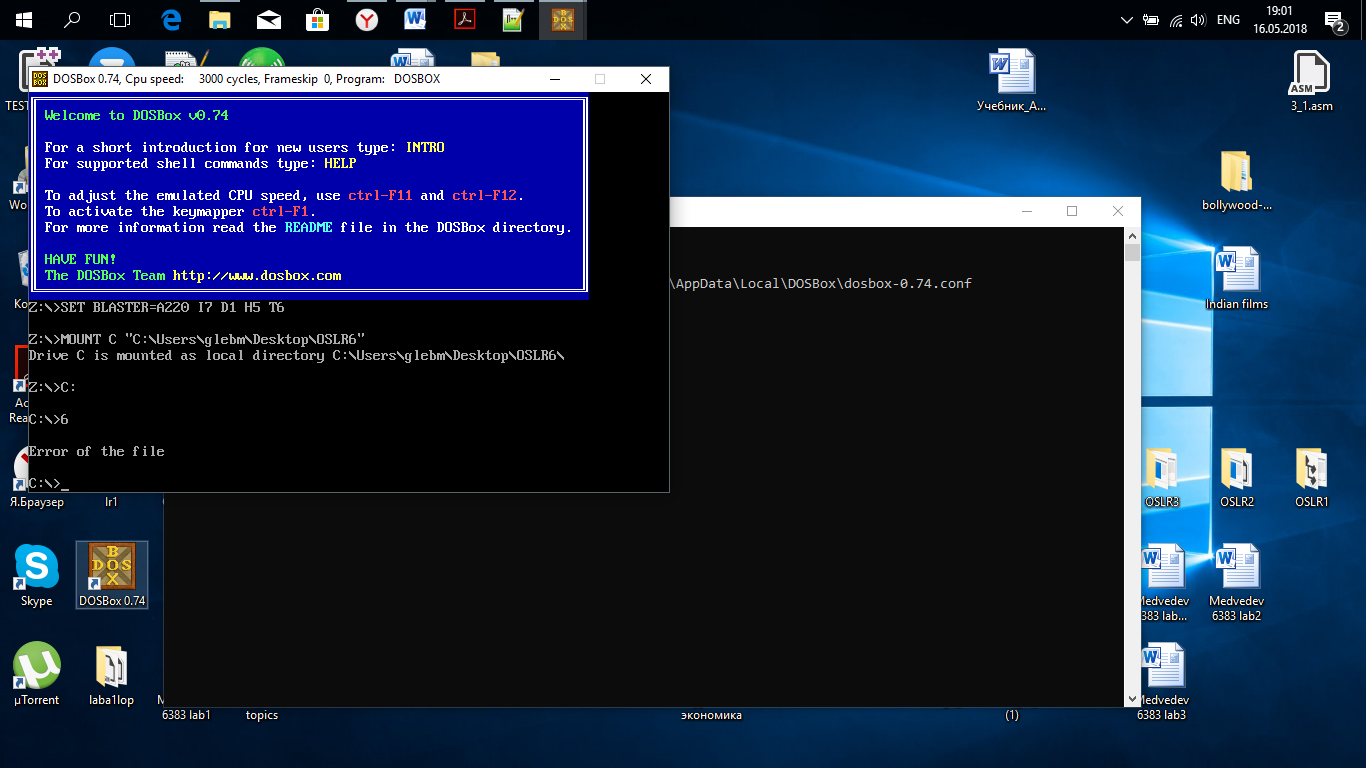


Рисунок 7 – Результат повторного выполнения программы 6.exe

**Выводы.**

В процессе выполнения данной лабораторной работы была исследована возможность построения загрузочного модуля динамической структуры.

**Ответы на контрольные вопросы.**

1. Как реализовано прерывание Ctrl+C?

При нажатии сочетания клавиш Ctrl+С, вызывается прерывание 23h, при этом управление передаётся по адресу 0000:008Ch. Данный адрес копируется в PSP функциями 26h и 4Ch, а затем восстанавливается при выходе из программы.

1. В какой точке заканчивается вызываемая программа, если код причины завершения 0?

В таком случае программа заканчивается в месте вызова функции 4Ch прерывания 21h.

1. В какой точке заканчивается вызываемая программа по прерыванию ctrl+c?

Программа заканчивается в месте вызова функции 01h прерывания 21h, то есть там, где ожидается ввод символа.

ПРИЛОЖЕНИЕ А

6.asm

CODE SEGMENT

ASSUME CS:CODE, DS:DATA, ES:DATA, SS:STACK

START: JMP BEGIN

;--------------------------

PRINT PROC near

mov AH,09h

int 21h

ret

PRINT ENDP

;--------------------------

TETR\_TO\_HEX PROC near

and al,0fh

cmp al,09

jbe NEXT

add al,07

NEXT: add al,30h

ret

TETR\_TO\_HEX ENDP

;--------------------------

BYTE\_TO\_HEX PROC near

push cx

mov ah,al

call TETR\_TO\_HEX

xchg al,ah

mov cl,4

shr al,cl

call TETR\_TO\_HEX ;

pop cx

ret

BYTE\_TO\_HEX ENDP

;--------------------------

MEMORY\_ PROC

mov ax,STACK

mov bx,es

sub ax,bx

add ax,10h

mov bx,ax

mov ah,4Ah

int 21h

jnc FIN

mov dx,offset ERRMEM

call PRINT

cmp ax,7

mov dx,offset MCB\_

je MEM\_PRINT

cmp ax,8

mov dx,offset NOTMEM

je MEM\_PRINT

cmp ax,9

mov dx,offset ERRADR

MEM\_PRINT:

call PRINT

mov dx,offset STRING

call PRINT

xor AL,AL

mov AH,4Ch

int 21H

FIN:

ret

MEMORY\_ ENDP

;--------------------------

PAR\_BL PROC

mov ax, es:[2Ch]

mov PARAM\_S,ax

mov PARAM\_S+2,es

mov PARAM\_S+4,80h

ret

PAR\_BL ENDP

;--------------------------

DEAL PROC

mov dx,offset STRING

call PRINT

mov dx,offset STD\_PATH

xor ch,ch

mov cl,es:[80h]

cmp cx,0

je UNTAIL

mov si,cx

push si

METOCHKA:

mov al,es:[81h+si]

mov [offset \_PATH+si-1],al

dec si

loop METOCHKA

pop si

mov [\_PATH+si-1],0

mov dx,offset \_PATH

UNTAIL:

push ds

pop es

mov bx,offset PARAM\_S

mov KEEP\_SP, SP

mov KEEP\_SS, SS

mov ax,4b00h

int 21h

jnc FIN\_

push ax

mov ax,DATA

mov ds,ax

pop ax

mov SS,KEEP\_SS

mov SP,KEEP\_SP

cmp ax,1

mov dx,offset ERRFUNCT

je DEAL\_PRINT

cmp ax,2

mov dx,offset ERRFILE

je DEAL\_PRINT

cmp ax,5

mov dx,offset ERRDISK

je DEAL\_PRINT

cmp ax,8

mov dx,offset NOTMEM\_

je DEAL\_PRINT

cmp ax,10

mov dx,offset ERRENV

je DEAL\_PRINT

cmp ax,11

mov dx,offset ERRFORM

DEAL\_PRINT:

call PRINT

mov dx,offset STRING

call PRINT

xor AL,AL

mov AH,4Ch

int 21H

FIN\_:

mov dx,offset STRING

call PRINT

mov ax,4d00h

int 21h

cmp ah,0

mov dx,offset GOOD

je REASONS

cmp ah,1

mov dx,offset END\_CTRL

je REASONS

cmp ah,2

mov dx,offset ERRDEVICE

je REASONS

cmp ah,3

mov dx,offset ERRRES

REASONS:

call PRINT

mov dx,offset STRING

call PRINT

mov dx,offset CODEELEM

call PRINT

call BYTE\_TO\_HEX

push ax

mov ah,02h

mov dl,al

int 21h

pop ax

xchg ah,al

mov ah,02h

mov dl,al

int 21h

mov dx,offset STRING

call PRINT

ret

DEAL ENDP

;--------------------------

BEGIN:

mov ax,DATA

mov ds,ax

call MEMORY\_

call PAR\_BL

call DEAL

xor AL,AL

mov AH,4Ch

int 21H

CODE ENDS

;--------------------------

DATA SEGMENT

ERRMEM db 'Error of clear memory: $'

MCB\_ db 'MCB is destroyed$'

NOTMEM db 'Not enough memory$'

ERRADR db 'Error of the addres$'

ERRFUNCT db 'Error of function number$'

ERRFILE db 'Error of the file$'

ERRDISK db 'Error of the disk$'

NOTMEM\_ db 'Not enough memory$'

ERRENV db 'Error of env$'

ERRFORM db 'Error of format$'

GOOD db 'All is good$'

END\_CTRL db 'End ctrl$'

ERRDEVICE db 'Error of device$'

ERRRES db 'End 31h$'

CODEELEM db 'Code of finish: $'

STRING db 0DH,0AH,'$'

PARAM\_S dw 0

dd ?

dd 0

dd 0

\_PATH db 50h dup ('$')

STD\_PATH db '2.COM',0

KEEP\_SS dw 0

KEEP\_SP dw 0

DATA ENDS

;--------------------------

STACK SEGMENT STACK

dw 64h dup (?)

STACK ENDS

END START