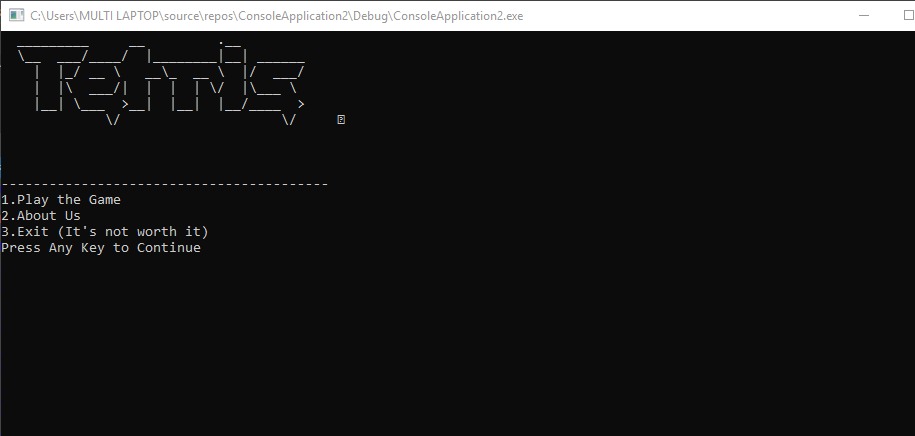
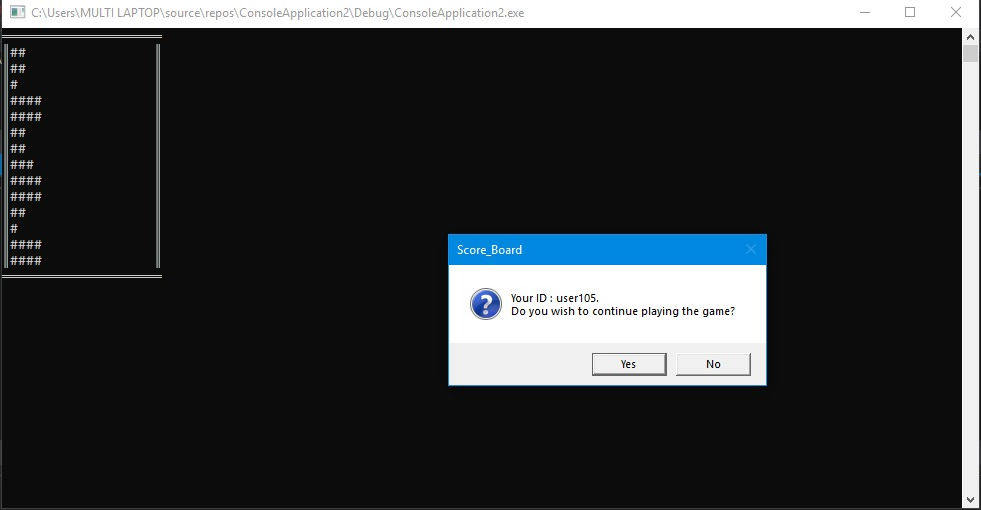
Final Project

**Project Name :** Tetris





Project Description:

For this Project, we will implement a code for game “Tetris”. Tetris is a puzzle matching strategy

Game. The game starts with tiles falling from above and the player has to place them accordingly.

Instructions to follow:

* **Press Any Key to start the game**
* **Secondly, in the black console screen**
* **Use the keys Z , X ,C, V ,B to move your shape across the border**
* **Finally create the stack and reach the end of the border.**
* **When you have reached the top most border. You score will be displayed in a message box.**

Code :

include irvine32.inc

;-------DATA SEGMENT----------;

.data

info1 db " \_\_\_\_\_\_\_\_\_ \_\_ .\_\_ "

info1\_size=($-info1)

db " \\_\_ \_\_\_/\_\_\_\_/ |\_\_\_\_\_\_\_\_|\_\_| \_\_\_\_\_\_ "

db " | |\_/ \_\_ \ \_\_\\_ \_\_ \ |/ \_\_\_/ "

db " | |\ \_\_\_/| | | | \/ |\\_\_\_ \ "

db " |\_\_| \\_\_\_ >\_\_| |\_\_| |\_\_/\_\_\_\_ > "

db " \/ \/ "

info1\_row dword 0

info1\_col dword 0

info2 db "-----------------------------------------",0

info3 db "1.Play the Game",0

info4 db "2.About Us ",0

info5 db "3.Exit (It's not worth it)",0

info6 db "Press Any Key to Continue",0

Array byte 20 dup (' ')

Rowsize = ($ - array)

byte 20 dup (' ')

byte 20 dup (' ')

byte 20 dup (' ')

byte 20 dup (' ')

byte 20 dup (' ')

byte 20 dup (' ')

byte 20 dup (' ')

byte 20 dup (' ')

byte 20 dup (' ')

byte 20 dup (' ')

byte 20 dup (' ')

byte 20 dup (' ')

byte 20 dup (' ')

captionW BYTE "Game Over",0

warningMsg BYTE "Congrats!!! You Got 15 points.Its a new Highscore "

BYTE "Previous Score : 9.",0

captionQ BYTE "Score\_Board",0

questionMsg BYTE "Your ID : user105."

BYTE 0dh,0ah,"Do you wish to continue playing the game?",0

remove\_counter db 0

esi\_count word 0

R\_index DWORD 0

C\_index DWORD 0

R\_count DWORD 14

num1 DWORD ?

shape\_1\_1 db "####",0

shape\_1\_2 db "####",0

shape\_2\_1 db "#",0

shape\_2\_2 db "###",0

shape\_3\_1 db "##",0

shape\_3\_2 db "##",0

inner\_sh db 0

outer\_sh db 0

tlc dd 0

save\_index db 100 dup(0)

save\_count dw 0

display\_count dw 0

row db 0

row2 db 0

column db 0

less db 0

count dw 0

temp dw ?

temp2 db ?

c\_shape db 0

key db ?

table\_size db ?

abc db ?

abc2 word ?

abc3 dword ?

abc4 db ?

end\_chk db ?

;-------CODE SEGMENT----------;

.code

main proc

;------ MAIN LOOP ----------;

call info\_layout

MOV EAX,0

MOV EBX,0

MOV ECX,0

MOV EDX,0

one\_time:

l1:

mov al,inner\_sh

mov abc4,al

call save

mov al,abc4

mov inner\_sh,al

mov R\_index , 0

mov C\_index , 0

mov R\_count , 14

mov eax,0

mov cl,end\_chk

cmp cl,1

je end\_msg

mov ecx,0

push edx

mov dl,0

mov dh,0

call gotoxy

pop edx

call readchar

mov key,ah

mov abc,ah

;mov inner\_sh,0

call DISPLAY

call dely

mov dl,0

mov dh,0

call gotoxy

cmp row,13

je l49

cmp less,0

je yx

mov dl,less

mov row,dl

mov dl,0

mov less,0

jmp l49

yx:

add row,2

JMP NXXT

l49:

push esi

push eax

mov esi,0

mov eax,0

mov si,save\_count

mov al,row

mov end\_chk,al

sub al,1

mov [save\_index+si],al

inc si

mov al,abc

mov [save\_index+si],al

inc si

mov al,inner\_sh

mov [save\_index+si],al

inc si

add save\_count,3

pop eax

pop esi

MOV ROW,0

add inner\_sh,8

cmp inner\_sh,24

jne nxxt

mov inner\_sh,0

NXXT:

jmp l1

end\_msg:

INVOKE MessageBox, NULL, ADDR warningMsg,

ADDR captionW,

MB\_OK + MB\_ICONEXCLAMATION

INVOKE MessageBox, NULL, ADDR questionMsg,

ADDR captionQ, MB\_YESNO + MB\_ICONQUESTION

cmp eax,IDYES

jz one\_time

exit

main endp

;---------Display------------;

DISPLAY proc

MOV temp,sp

mov esi,offset Array

mov ecx ,14

mov ebx ,0

;----- ROW COUNT ------- ;

l1:

mov num1,ecx

mov eax,R\_index

cmp eax,0

jne cont

first\_line:

mov ecx,20

fl:

push ax

mov al,0cdh

call writechar

pop ax

loop fl

call crlf

cont:

mov ecx,R\_count

mul ecx

mov ebx ,eax

mov C\_index,0

add esi ,ebx

mov ecx,20

mov eax,R\_index

mov remove\_counter,0

;----- COLUMN COUNT ----- ;

;proc

lxd:

push ebx

push esi

push eax

mov eax,0

mov ebx,0

mov eax,rowsize

mov ebx,R\_index

mul bx

MOV EBX, OFFSET array

ADD EBX,eax

MOV ESI, c\_index

MOV AL,[EBX+ESI]

pop esi

pop ebx

cmp al,23h

jne in\_count

inc remove\_counter

in\_count:

loop lxd

push edx

mov dl,remove\_counter

cmp dl,12

je spaces

;proc end

mov ecx,20

l2:

ADD esi ,C\_index

mov edx ,C\_index

push dx

cmp edx,0

je vertical

cmp edx,19

je vertical

mov dl,row

cmp edx,R\_index

je l55

;------PRINT SPACES-------;

xyz:

push ebx

push esi

push eax

mov eax,0

mov ebx,0

mov eax,rowsize

mov ebx,R\_index

mul bx

MOV EBX, OFFSET array

ADD EBX,eax

MOV ESI, c\_index

MOV AL,[EBX+ESI]

pop esi

pop ebx

cmp al,20h

je l123

cmp al,23h

je l124

jmp top

l123:

mov al,' '

call writechar

jmp top

l124:

mov edx,c\_index

cmp edx,15h

jae top

mov al,'#'

call writechar

inc remove\_counter

top:

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

push edx

mov dl,remove\_counter

cmp dl,12

je spaces

jne nZZ

spaces:

l96:

push ecx

mov ecx,15

l97:

push ebx

push esi

push eax

mov eax,0

mov ebx,0

mov eax,rowsize

mov ebx,R\_index

mul bx

MOV EBX, OFFSET array

ADD EBX,eax

MOV ESI,ecx

mov al,' '

MOV [EBX+ESI] ,al

pop eax

pop esi

pop ebx

loop l97

pop ecx

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

nzz:

pop edx

inc C\_index

dec ecx

cmp ecx,0

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

jne l2

call crlf

inc R\_index

mov ecx ,num1

dec ecx

cmp ecx,0

jne l1

;---------END Loop L1------------;

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

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

mov ecx,20

horizontal:

push ax

mov al,0cdh

call writechar

pop ax

loop horizontal

jmp down

vertical:

push ax

mov al,0bah

call writechar

pop ax

jmp top

; 1-4 left

; 5-9 mid

; 10-14 right

l55:

pop dx

push ax

mov eax,0

mov al,key

cmp al,2Eh

jne l\_r

;-----SHAPE TO THE MID -------;

cmp dx,8

mov column,8

je l66

jmp xyz

l\_r:

cmp al, 30H

je right

cmp al,2Ch

je left

cmp al,2Fh

je right\_2

cmp al,2DH

je left\_2

jmp xyz

;-----SHAPE TO THE RIGHT -------;

right:

cmp edx,14

mov column,14

je l66

jmp xyz

;-----SHAPE TO THE LEFT -------;

left:

cmp edx,1

mov column,1

je l66

jmp xyz

;-----SHAPE TO THE RIGHT\_2 -------;

right\_2:

cmp edx,11

mov column,11

je l66

jmp xyz

;-----SHAPE TO THE LEFT\_2 -------;

left\_2:

cmp edx,4

mov column,4

je l66

jmp xyz

l66:

push ebx

push esi

push eax

mov eax,0

mov ebx,0

mov eax,rowsize

mov ebx,R\_index

add ebx,1

mul bx

MOV EBX, OFFSET array

ADD EBX,eax

MOV ESI, c\_index

MOV AL,[EBX+ESI]

pop esi

pop ebx

cmp al,23h

je shal

jmp chal

chal:

call shape

add row,1

pop ax

jmp xyz

down:

MOV sp,temp

ret

shal:

mov edX,R\_index

mov less,dl

mov dl,0

jmp down

DISPLAY ENDP

;---------Display END------------;

;---------Shape------------;

shape proc

cmp inner\_sh ,0

je sh1

cmp inner\_sh ,8

je sh2

cmp inner\_sh ,16

je sh3

sh1:

cmp outer\_sh,1

je sh12

push edx

mov edx,offset shape\_1\_1

call writestring

inc outer\_sh

pop edx

add C\_index,4

ret

sh12:

push edx

mov edx,offset shape\_1\_2

call writestring

pop edx

add row,0FFh

mov key,0ffh

dec outer\_sh

add C\_index,4

ret

sh2:

cmp outer\_sh,1

je sh21

push edx

mov edx,offset shape\_2\_1

call writestring

inc outer\_sh

pop edx

add C\_index,1

ret

sh21:

push edx

mov edx,offset shape\_2\_2

call writestring

pop edx

mov key,0ffh

dec outer\_sh

add row,0FFh

add C\_index,3

ret

sh3:

cmp outer\_sh,1

je sh31

push edx

mov edx,offset shape\_3\_1

call writestring

inc outer\_sh

pop edx

add C\_index,2

ret

sh31:

push edx

mov edx,offset shape\_3\_2

call writestring

pop edx

mov key,0ffh

dec outer\_sh

add row,0FFh

add C\_index,2

ret

shape endp

;---------Shape END------------;

;---------Delay------------;

dely proc

delRep:

push cx

mov cx, 5FFFH

delDec:

dec cx

jnz delDec

pop cx

dec cx

jnz delRep

ret

dely endp

;---------Delay END------------;

;---------Shape END------------;

;---------Delay------------;

;----------display2-----------

DISPLAY\_2 proc

MOV temp,sp

mov esi,offset Array

mov ecx ,14

mov ebx ,0

;----- ROW COUNT ------- ;

l1:

mov num1,ecx

mov eax,R\_index

cmp eax,0

jne cont

first\_line:

mov ecx,20

fl:

push ax

mov al,0cdh

call writechar

pop ax

loop fl

call crlf

cont:

mov ecx,R\_count

mul ecx

mov ebx ,eax

mov C\_index,0

add esi ,ebx

mov ecx,20

mov eax,R\_index

;----- COLUMN COUNT ----- ;

mov remove\_counter,0

l2:

ADD esi ,C\_index

mov edx ,C\_index

push dx

cmp edx,0

je vertical

cmp edx,19

je vertical

mov dl,row

cmp edx,R\_index

je l55

;------PRINT SPACES-------;

xyz:

push ebx

push esi

push eax

mov eax,0

mov ebx,0

mov eax,rowsize

mov ebx,R\_index

mul bx

MOV EBX, OFFSET array

ADD EBX,eax

MOV ESI, c\_index

MOV AL,[EBX+ESI]

pop ebx

cmp al,20h

je l123

cmp al,23h

je l124

jmp top

l123:

mov al,' '

call writechar

jmp top

l124:

mov edx,c\_index

cmp edx,15h

jae top

mov al,'#'

call writechar

inc remove\_counter

;------MOV FROM COLUNMNS THEN CHANGE ROW AND CALL LOOP L1-------;

top:

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

push edx

mov dl,remove\_counter

cmp dl,12

je spaces

jne nZZ

spaces:

l96:

push ecx

mov ecx,15

l97:

push ebx

push esi

push eax

mov eax,0

mov ebx,0

mov eax,rowsize

mov ebx,R\_index

mul bx

MOV EBX, OFFSET array

ADD EBX,eax

MOV ESI,ecx

mov al,' '

MOV [EBX+ESI] ,al

pop eax

pop esi

pop ebx

loop l97

pop ecx

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

nzz:

pop edx

inc C\_index

dec ecx

cmp ecx,0

jne l2

call crlf

inc R\_index

mov ecx ,num1

dec ecx

cmp ecx,0

jne l1

;---------END Loop L1------------;

mov ecx,20

horizantal:

push ax

mov al,0cdh

call writechar

pop ax

loop horizantal

jmp down

vertical:

push ax

mov al,0bah

call writechar

pop ax

jmp top

; 1-4 left

; 5-9 mid

; 10-14 right

l55:

pop dx

push ax

mov eax,0

mov al,key

cmp al,2Eh

jne l\_r

;-----SHAPE TO THE MID -------;

cmp dx,8

mov column,8

je l66

jmp xyz

l\_r:

cmp al, 30H

je right

cmp al,2Ch

je left

cmp al,2Fh

je right\_2

cmp al,2DH

je left\_2

jmp xyz

;-----SHAPE TO THE RIGHT -------;

right:

cmp edx,14

mov column,14

je l66

jmp xyz

;-----SHAPE TO THE LEFT -------;

left:

cmp edx,1

mov column,1

je l66

jmp xyz

;-----SHAPE TO THE RIGHT\_2 -------;

right\_2:

cmp edx,11

mov column,11

je l66

jmp xyz

;-----SHAPE TO THE LEFT\_2 -------;

left\_2:

cmp edx,4

mov column,4

je l66

jmp xyz

l66:

call shape2

add row,1

pop ax

jmp xyz

down:

MOV sp,temp

ret

DISPLAY\_2 ENDP

save proc

mov R\_index , 0

mov C\_index , 0

mov R\_count , 14

mov al,row

mov row2,al

mov eax,0

mov cx,save\_count

mov esi\_count,0

cmp cx,0

je l91

l90:

mov R\_index , 0

mov C\_index , 0

mov R\_count , 14

mov esi,0

mov si,esi\_count

mov al,[save\_index+si]

mov row,al

inc si

dec cx

mov al,[save\_index+si]

mov key,al

inc si

dec cx

mov al,[save\_index+si]

mov inner\_sh,al

dec cx

inc si

mov abc2,cx

mov esi\_count,si

push edx

mov dl,0

mov dh,0

call gotoxy

pop edx

call display\_2

mov cx,abc2

cmp ecx,0

mov si,esi\_count

jne l90

mov al,temp2

mov row,al

mov column,0

mov key,0

l91:

mov al,row2

mov row,al

ret

save endp

;----------------shape2--------------

shape2 proc

cmp inner\_sh ,0

je sh1

cmp inner\_sh ,8

je sh2

cmp inner\_sh ,16

je sh3

sh1:

cmp outer\_sh,1

je sh12

push ebx

push esi

push eax

mov eax,0

mov ebx,0

mov eax,rowsize

mov ebx,R\_index

mul bx

MOV EBX, OFFSET array

ADD EBX,eax

MOV ESI, c\_index

mov al,'#'

MOV [EBX+ESI],al

inc esi

MOV [EBX+ESI],al

inc esi

MOV [EBX+ESI],al

inc esi

MOV [EBX+ESI],al

cmp outer\_sh,1

pop eax

pop esi

pop ebx

push edx

mov edx,offset shape\_1\_1

call writestring

inc outer\_sh

pop edx

add C\_index,4

ret

sh12:

push ebx

push esi

push eax

mov eax,0

mov ebx,0

mov eax,rowsize

mov ebx,R\_index

mul bx

MOV EBX, OFFSET array

ADD EBX,eax

MOV ESI, c\_index

mov al,'#'

MOV [EBX+ESI],al

inc esi

MOV [EBX+ESI],al

inc esi

MOV [EBX+ESI],al

inc esi

MOV [EBX+ESI],al

cmp outer\_sh,1

pop eax

pop esi

pop ebx

push edx

mov edx,offset shape\_1\_2

call writestring

pop edx

add row,0FFh

mov key,0ffh

dec outer\_sh

add C\_index,4

ret

sh2:

push ebx

push esi

push eax

mov eax,0

mov ebx,0

mov eax,rowsize

mov ebx,R\_index

mul bx

MOV EBX, OFFSET array

ADD EBX,eax

MOV ESI, c\_index

mov al,'#'

MOV [EBX+ESI],al

cmp outer\_sh,1

pop eax

pop esi

pop ebx

cmp outer\_sh,1

je sh21

push edx

mov edx,offset shape\_2\_1

call writestring

inc outer\_sh

pop edx

add C\_index,1

ret

sh21:

push ebx

push esi

push eax

mov eax,0

mov ebx,0

mov eax,rowsize

mov ebx,R\_index

mul bx

MOV EBX, OFFSET array

ADD EBX,eax

MOV ESI, c\_index

mov al,'#'

MOV [EBX+ESI],al

inc esi

MOV [EBX+ESI],al

inc esi

MOV [EBX+ESI],al

cmp outer\_sh,1

pop eax

pop esi

pop ebx

push edx

mov edx,offset shape\_2\_2

call writestring

pop edx

mov key,0ffh

dec outer\_sh

add row,0FFh

add C\_index,3

ret

sh3:

push ebx

push esi

push eax

mov eax,0

mov ebx,0

mov eax,rowsize

mov ebx,R\_index

mul bx

MOV EBX, OFFSET array

ADD EBX,eax

MOV ESI, c\_index

mov al,'#'

MOV [EBX+ESI],al

inc esi

MOV [EBX+ESI],al

pop eax

pop esi

pop ebx

cmp outer\_sh,1

je sh31

push edx

mov edx,offset shape\_3\_1

call writestring

inc outer\_sh

pop edx

add C\_index,2

ret

sh31:

push ebx

push esi

push eax

mov eax,0

mov ebx,0

mov eax,rowsize

mov ebx,R\_index

mul bx

MOV EBX, OFFSET array

ADD EBX,eax

MOV ESI, c\_index

mov al,'#'

MOV [EBX+ESI],al

inc esi

MOV [EBX+ESI],al

pop eax

pop esi

pop ebx

push edx

mov edx,offset shape\_3\_2

call writestring

pop edx

mov key,0ffh

dec outer\_sh

add row,0FFh

add C\_index,2

ret

shape2 endp

;----------info\_layout-------------;

info\_layout PROC

mov ecx,6

layout1:

mov ebx,offset info1

mov eax,0

mov edx,0

mov eax,info1\_size

mov edx,info1\_row

mul edx

add ebx,edx

mov esi,info1\_col

MOV al,[ebx+esi]

call writechar

push ecx

mov ecx,42

layout2:

inc info1\_col

mov esi,info1\_col

MOV al,[ebx+esi]

call writechar

loop layout2

call crlf

pop ecx

inc info1\_row

LOOP layout1

MOV ecx,3

l\_space:

call crlf

Loop l\_space

mov edx,offset info2

call writestring

call crlf

mov edx,offset info3

call writestring

call crlf

mov edx,offset info4

call writestring

call crlf

mov edx,offset info5

call writestring

call crlf

mov edx,offset info6

call writestring

call crlf

call readchar

call clrscr

ret

info\_layout ENDP

;------info\_layout ENDP------------;

end main