המשחק

# תנאים לולאות ופעולות

## bg11

במשחק שלנו אם הדמות נוגעת בקצוות המחודדים שבמסגרת המחשק השחקן נפסל.

לצורך כך יש לבדוק האם במיקום הבא אליו השחקן זז יש את הסימנים < או > או v או ^.

אם שחקן נוגע באחד מהתווים האלו יש פסילה אחרת הוא יכול להתקדם ולהמשיך לשחק.

**אך מה קורה אם נגענו בדופן הלא משונן של לוח המשחק?**

* אם נגענו בדופן הלא משונן של גבולות לוח המשחק נרצה שהשחקן שלנו **יחליף כיוון**, במקום לזוז שמאלה יזוז ימינה ולהפך.
* תזוזה של הדמות לכיוון שמאל ⇨ מוסיפים למשתנה המייצג את המיקום של הסמן על ציר ה – x.
* תזוזה של הדמות לכיוון ימין ⇦ מפחיתים מהשתנה המייצג את המיקום של הסמן על ציר ה – x.

## פעולה המשנה את כיוון התזוזה של השחקן

ניצור משתנה בשם direction אשר יקבל את הערכים (ימין) **r**, או (שמאל) **l**.

נאתחל את המשתנה בערך r

כי הדמות שלנו נעה באלכסון ימינה ומטה.

direction db 'r'

## פעולה המשנה את כיוון התזוזה של השחקן – changeDir

* **אלגוריתם:**

האם [direction] = ‘r’

אם כן (je) קפוץ לתווית **left**

[direction] = ‘r’

[x\_cord] + 2 = [x\_cord]

קפוץ לתווית **endChangeDir**

תווית: **left**

[direction] = ‘l’

[x\_cord] -2 = [x\_cord]

תווית: **endChangeDir**

זימון הפעולה למיקום הסמן

זימון הפעולה למיקום הסמן - setCursorePosition

; change character directionfro 'r' to 'l'and from 'l' to 'r'

proc changeDir

pusha

cmp [direction], 'r'

je left

mov [direction], 'r'

add [x\_cord], 2

jmp endChangeDir

left:

mov [direction], 'l'

sub [x\_cord], 2

endChangeDir:

; sets character position in opposite direction

**call setCursorePosition**

popa

ret

endp changeDir

זימון הפעולה למיקום הסמן – setCursorePosition

עבר מהלולאה הראשית לתוך הפעולה הקובעת את כיוון תזוזת השחקן.

שימו לב 🎔, יש לנו זימון של פעולה מתוך פעולה.

באסמבלי מותר לזמן פעולה מתוך פעולה, בתנאי שלא קופצים לתווית בפעולה אחרת.

יש לצאת מכל פעולה בצורה מסודרת, בכל פועלה מותר לקפוץ לתוויות בתוך הפעולה. אך בסוף קופצים לסוף הפעולה ויוצרים בעזרת פקודת ret וחוזרים לפועלה המזמנת ושוב יוצאים ממנה בצורה מסודרת.

**תזוזת הדמות במהלך המשחק**

* עד עתה הדמות שלנו נעה על לוח המחשק ימינה ומטה לכן התזוזה של השחקן הייתה בהוספת 2 למשתנה המייצג את מיקום הסמן על ציר ה – x.
* add [y\_cord], 2
* עתה, נרצה להוסיף או להפחית מהמשתנה המיצג את מיקום הסמן על ציר ה – x
* בהתאם לכיוון שקבענו בפעולה המשנה את כיוון התנועה של השחקן.
* לכן במקום הפקודה של הוספה 2 למשתנה [x\_cord]
* ניצור פעולה שבה התנאים:

אם כיוון התנועה = r נגדיל את המשתנה [x\_cord]

אם כיוון התנועה = l נקטין את המשתנה [x\_cord]

אם הקשנו על מקש רווח = נקטין את המשתנה [y\_cord]

## פעולה המחשבת את תזוזת השחקן

האם [direction] = **r** (הכיוון ימינה ⇨)

אם כן קפוץ לתווית - **right**

הפחת 1 מ – [x\_cord] (הכיוון שמאלה ⇦)

קפוץ לתווית - **calc\_y**

* **תווית – right**

הגדל את [x\_cord] ב – 1

**תווית - calc\_y**

האם [chr] = ‘ ‘ (האם הקשנו על מקש רווח במקלדת)

אם כן קפוץ לתווית - **jmpChr**

הגדל את [y\_cord] ב – 2 (הכיוון ⇩)

קפוץ לתווית: **endCalculateCords**

**תווית jmpChr**

הפחת 2 מ – [y\_cord] (הכיוון ⇧)

תווית **endCalculateCords**

; increases x\_cord by one if direction is 'r' and decreases it if it is 'l'

; increases y\_cord by 2 if chr is not ' ' and decreases it by 1 if it is

proc calculateCords

pusha

; increase x\_cord by 1

cmp [direction], 'r'

je right

dec [x\_cord]

jmp calc\_y

right:

inc [x\_cord]

calc\_y:

cmp [chr], ' '

je jmpChr

add [y\_cord], 2

jmp endCalculateCords

jmpChr:

sub [y\_cord], 2

endCalculateCords:

popa

ret

endp calculateCords

## האם נגענו בדופן הלא "מחודד" של גבולות לוח המשחק?



* עלינו לשנות את הפעולה הבודקת אם הגענו ל"קצה מחודד"
* ולהוסיף את התנאי האם הגענו למיקום בו יש את התו ‘|’
* (המסמן את דופן לוח המשחק).
* מתוך פולה זו נזמן את הפעולה הקובעת את כיוון תזוזת הדמות.

## נוסיף לפעולה בדקיה האם נגענו בדופן לוח המשחק?

; check the cotent of scrrenChe.

; if it is one of <>^v ends game

## הפעולה האם נגענו ב"שפיץ"

; check the content of screenChr

; if it is one of <>^v ends game

proc checkScrnChr

pusha

mov [lost], 0

cmp [screenChr], '<'

je youLost

cmp [screenChr], '>'

je youLost

cmp [screenChr], 86 ; V

je youLost

cmp [screenChr], 94 ; ^

je youLost

jmp endCheckScrnChr

youLost:

mov [lost], 1

endCheckScrnChr:

popa

ret

endp checkScrnChr

proc checkScrnChr

pusha

cmp [screenChr], '>'

je youLost

cmp [screenChr], '<'

je youLost

cmp [screenChr], 86 ; V

je youLost

cmp [screenChr], 94 ;^

je youLost

**cmp [screenChr], '|'**

**je changeDirection**

jmp endCheckScrnChr

youLost:

mov [lost], 1

jmp endCheckScrnChr

**changeDirection:**

**call changeDir**

endCheckScrnChr:

popa

ret

endp checkScrnChr

## הלולאה של התכנית הראשית

* איפוס המשתנה השומר את התו האחרון שהוקש [chr]
* מחיקת השחקן
* בהתאם למשתנה של הכיוון [direction] - קביעת המשתנים של מיקום הסמן
* מיקום הסמן על המסך
* קריאה של התו במיקום הסמן וקיבעת הערך של משתנה [lost]
* בדיקת האם [lost] = פסילה
* אם כן קפיצה לסןף המשחק
* ציור שחקן במיקום הסמן
* בדיקת איזה תו הוקש במקלדת והשמה ל – [chr]

## הלולאה הראשית של המשחק:

mov [chr], 0

mainGameLoop:

; delete character

; move charcter by 1 step

mov [color], 0

call drawCharacter

בודק את כיוון התנועה ע"י משתנה [direction] וקובע את תנועת הסמן לנקודה החדשה.

call calculateCords

; print character at cursor location

call setCursorePosition

call readScreenChr

call checkScrnChr

cmp [lost], 1

je end\_game

; draw smiley - ascii 2 at cursor position

call drawCharacter

; check if thre is a charcter to read

mov [chr], 0

mov ah, 1h

int 16h

jz noKey

; read a chararcter

call readChr

; check if user asks to quit

cmp [chr], 'q'

je end\_game

noKey:

call delay

jmp mainGameLoop

end\_game:

; text mode

mov ax, 2h

int 10h

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

exit:

mov ax, 4c00h

int 21h

## הפסיקות בתכנית שלנו

|  |  |
| --- | --- |
| פקודה להדפסת מחרוזת  (שם משתנה המחרוזת msg) | mov dx, offset msg  mov ah, 9h  int 21h |
| פקודה לקליטת תו מהמקלדת  לא מדפיסה את תו ולא ממשיכה בהוראות על לקבלת תו. | mov ah, 0h  int 16h  הפסיקה מחזירה בתוךal את קוד ה־ ASCII של התו שנמצא בראש הבאפר ובתוךah את ה־code scan שלו. בנוסף, הפסיקה "מנקה" את התו מהבאפר |
| פקודה לקליטת תו מהמקלדת  לא מדפיסה את תו, אם לא הוקש על מקש ממשיכה בפקודות התכנית. | mov ah, 1h  int 16h  הפסיקה מדליקה את ה zero flag– אם יש תו מוכן לקריאה (zf = 1) , ומכבה אם אין תו מוכן (zf = 0).  אם יש תו מוכן, al ו-ah יקבלו את ערכי ה־ASCII וה־code scan של התו. הפסיקה אינה "מנקה" את התו מהבאפר |
| כניסה לתצוגה גרפית  25\*40 | mov ax, 13h  int 10h |
| יציאה מתצוגה גרפית  80\*25 | mov ax, 2h  int 10h |
| מיקום הסמן על המסך | mov dh, [y\_cord] ; row  mov dl, [x\_cord] ; column  mov bh, 0 ; page number  mov ah, 2  int 10h |
| ציור תו על המסך במיקום הסמן | mov ah, 9  mov al, 2 ; al = character to display  mov bx, [color] ; bh = Background bl = Foreground  mov cx, 1 ; cx = number of times to write character  int 10h |
| השהייה  המתנה של פרק זמן קצר (חמישית השנייה) | mov cx, 03h ;High Word  mov dx, 4240h ;Low Word  mov al, 0  mov ah, 86h ;Wait  int 15h |
| קריאה של תו במיקום הסמן.  שמירה התו למשתנה  [screenChr] | mov bl, 0h ; Page=1  mov ah, 08h ; Read character function  int 10h ;return the character to al  mov [screenChr], al |

**INT 16h / AH = 00h** - get keystroke from keyboard (no echo).

return:

AH = BIOS scan code.

AL = ASCII character.

(if a keystroke is present, it is removed from the keyboard buffer).

**INT 16h / AH = 01h** - check for keystroke in the keyboard buffer.

return:

ZF = 1 if keystroke is not available. ⇨ לא הוקש מקש

ZF = 0 if keystroke available. ⇨ הוקש מקש

AH = BIOS scan code.

AL = ASCII character.

(if a keystroke is present, it is not removed from the keyboard buffer

Back to Top).

**INT 15h** / **AH = 86h** - BIOS wait function.

**CX:DX** = interval in microseconds

Note: 1 million in hex is '000f 4240',

so CX = 0fh : DX = 4240h = 1 second

**CX = 3h : DX = 4240H = 1/5 second**

תרגול

## סיכום קצר

למדנו:

הדפסה למסך – פלט

קליטת תווים מהמקלדת – קלט

קריאת תווים מהמסך - פלט

משתנים

פעולות לוגיות (חיבור וחיסור)

לולאות

תנאים ופעולות

במשחק שלנו:

הדפסה של מסך הפתיחה

הדפסה של שם יוצר המשחק

המתנה לתו

הדפסה של לוח המשחק

המתנה לתו

**לולאה ראשית**

מחיקת שחקן

הזזת השחקן על פני לוח המשחק

בדיקה האם הקשנו על תו רווח

אם כן - שינוי משתנה הקובע את השחקן על ציר y מיקום סמן

מיקום סמן

קריאה של "באפר" המקלדת

אם הקשנו על q יציאה מהמשחק

ציור השחקן

בדיקת האם הוקש מקש במקלדת.

במשחק שלנו אם הדמות נוגעת בקצוות המחודדים שבמסגרת המחשק השחקן נפסל.

לצורך כך יש לבדוק האם במיקום הבא אליו השחקן זז יש את הסימנים < או > או v או ^.

אם שחקן נוגע באחד מהתווים האלו יש פסילה אחרת הוא יכול להתקדם ולהמשיך לשחק.

שמרו את התכנית שכתבתם בשיעור קודם בשם חדש והמשיכו לעבוד בקובץ החדש.

## מה קורה אם נגענו בדופן הלא משונן של לוח המשחק?

אם נגענו בדופן הלא משונן של גבולות לוח המשחק נרצה שהשחקן שלנו **יחליף כיוון**, במקום לזוז שמאלה יזוז ימינה ולהפך.

תזוזה של הדמות לכיוון שמאל ⇨ מוסיפים למשתנה המייצג את המיקום של הסמן על ציר ה – x [x\_cord].

תזוזה של הדמות לכיוון ימין ⇦ מפחיתים מהשתנה המייצג את המיקום של הסמן על ציר ה – x [x\_cord].

## פעולה המשנה את כיוון התזוזה של השחקן

ניצור משתנה בשם direction אשר יקבל את הערכים (ימין) **r**, או (שמאל) **l**.

נאתחל את המשתנה בערך r

כי הדמות שלנו נעה באלכסון ימינה ומטה.

במקטע הנתונים:

direction db 'r'

נאתחל את הערך ל – r (right) כי השחקן שלנבו מתחיל לנוע ימינה.

**הפעולה המשנה את כיון התזוזה של השחקן:**

|  |  |
| --- | --- |
| **אלגוריתים:**  האם [direction] = ‘r’  אם כן (je) קפוץ לתווית **left**  [direction] = ‘r’  [x\_cord] + 2 = [x\_cord]  קפוץ לתווית **endChangeDir**  תווית: **left**  [direction] = ‘l’  [x\_cord] -2 = [x\_cord]  תווית: **endChangeDir**  זימון הפעלה למיקום הסמן | ; change character directionfro 'r' to 'l'and from 'l' to 'r'  proc changeDir  pusha  cmp [direction], 'r'  je left  mov [direction], 'r'  add [x\_cord], 2  jmp endChangeDir  left:  mov [direction], 'l'  sub [x\_cord], 2    endChangeDir:  ; sets character position in opposite direction  **call setCursorePosition**  popa  ret  endp changeDir |

זימון הפעולה למיקום הסמן – setCursorePosition

עבר מהלולאה הראשית לתוך הפעולה הקובעת את כיוון תזוזת השחקן.

שימו לב 🎔, יש לנו זימון של פעולה מתוך פעולה.

באסמבלי מותר לזמן פעולה מתוך פעולה, בתנאי שלא קופצים לתווית בפעולה אחרת.

נמחקן מתוך הלולאה הראשית את פעולה הממקמת את הסמן

ובמקומה נזמן את הפעולה הקובעת את כיוון התנועה של השחקן.

**תזוזת הדמות במהלך המשחק**

עד עתה הדמות שלנו נעה על לוח המחשק ימינה ומטה לכן התזוזה של השחקן הייתה בהוספת 1 למשתנה המייצג את מיקום הסמן על ציר ה – x [x\_cord]

inc [x\_cord]

הוספת 2 למשתנה המייצג את מיקום הסמן על ציר ה – y [y\_cord]

add [y\_cord], 2

עתה, נרצה להוסיף או להפחית מהמשתנה המיצג את מיקום הסמן על ציר ה – x

בהתאם לכיוון שקבענו בפעולה המשנה את כיוון התנועה של השחקן.

לכן במקום הפקודה של הוספה 2 למשתנה [x\_cord]

ניצור פעולה שבה התנאים:

אם כיוון התנועה = r נגדיל את המשתנה [x\_cord]

אם כיוון התנועה = l נקטין את המשתנה [x\_cord]

אם הקשנו על מקש רווח = נקטין את המשתנה [y\_cord]

## פעולה המחשבת את תזוזת השחקן

האם [direction] = **r** (הכיוון ימינה ⇨)

אם כן קפוץ לתווית - **right**

הפחת 1 מ – [x\_cord] (הכיוון שמאלה ⇦)

קפוץ לתווית - **calc\_y**

* **תווית – right**

הגדל את [x\_cord] ב – 1

**תווית - calc\_y**

האם [chr] = ‘ ‘ (האם הקשנו על מקש רווח במקלדת)

אם כן קפוץ לתווית - **jmpChr**

הגדל את [y\_cord] ב – 2 (הכיוון ⇩)

קפוץ לתווית: **endCalculateCords**

**תווית jmpChr**

הפחת 2 מ – [y\_cord] (הכיוון ⇧)

תווית **endCalculateCords**

; increases x\_cord by one if direction is 'r' and decreases it if it is 'l'

; increases y\_cord by 2 if chr is not ' ' and decreases it by 1 if it is

proc calculateCords

pusha

; increase x\_cord by 1

cmp [direction], 'r'

je right

dec [x\_cord]

jmp calc\_y

right:

inc [x\_cord]

calc\_y:

cmp [chr], ' '

je jmpChr

add [y\_cord], 2

jmp endCalculateCords

jmpChr:

sub [y\_cord], 2

endCalculateCords:

popa

ret

endp calculateCords

## האם נגענו בדופן הלא "מחודד" של גבולות לוח המשחק?



* עלינו לשנות את הפעולה הבודקת אם הגענו ל"קצה מחודד"
* ולהוסיף את התנאי האם הגענו למיקום בו יש את התו ‘|’
* (המסמן את דופן לוח המשחק).
* מתוך פולה זו נזמן את הפעולה הקובעת את כיוון תזוזת הדמות.

## פעולה הבודקת האם נגענו ב"שפיץ"

; check the content of screenChr.

; if it is one of <>^v ends game

proc checkScrnChr

pusha

mov [lost], 0

cmp [screenChr], '<'

je youLost

cmp [screenChr], '>'

je youLost

cmp [screenChr], 86 ; V

je youLost

cmp [screenChr], 94 ; ^

je youLost

jmp endCheckScrnChr

youLost:

mov [lost], 1

endCheckScrnChr:

popa

ret

endp checkScrnChr

## נוסיף לפעולה בדקיה האם נגענו בדופן לוח המשחק?

; check the cotent of scrrenChe.

; if it is one of <>^v ends game

proc checkScrnChr

pusha

cmp [screenChr], '>'

je youLost

cmp [screenChr], '<'

je youLost

cmp [screenChr], 86 ; V

je youLost

cmp [screenChr], 94 ;^

je youLost

**cmp [screenChr], '|'**

**je changeDirection**

jmp endCheckScrnChr

youLost:

mov [lost], 1

jmp endCheckScrnChr

**changeDirection:**

**call changeDir**

endCheckScrnChr:

popa

ret

endp checkScrnChr

הלולאה הראשית של המשחק:

; increase x\_cord by 1

inc [x\_cord]

cmp [chr], ' '

je jmpCar

add [y\_cord], 2

jmp continue

jmpCar:

sub [y\_cord], 2

continue:

מאחר והכנסנו את הבדיקה של האם הקשנו על מקש הרווח לפעולה calculateCords

הבודקת מהו הכיוון של תנועת השחקן.

עלינו למחוק את הפקודות מהתכנית הראשית:

mov [chr], 0

**mainGameLoop:**

; delete character

mov [color], 0

call drawCharacter

בודק את כיוון התנועה ע"י משתנה [direction] וקובע את תנועת הסמן לנקודה החדשה.

; set cursor location and direction

call calculateCords

call setCursorePosition

; check character at cursor location

call readScreenChr

call checkScrnChr

cmp [lost], 1

je end\_game

; draw smiley - ascii 2 at cursor position

mov [color], 0Eh

call drawCharacter

; check if thre is a charcter to read

mov [chr], 0

mov ah, 1h

int 16h

jz noKey

; read a character

call readChr

; check if user asks to quit

cmp [chr], 'q'

je end\_game

noKey:

call delay

jmp mainGameLoop

end\_game:

; text mode

mov ax, 2h

int 10h

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

exit:

mov ax, 4c00h

int 21h

**Loop forever**

נהפוך את הלולאה ללולאת לעולמים:

נמחק את מונה הלולאה mov cx, 20

לפני התווית של הלולאה הראשית: mainGameLoop

בסוף הלולאה נשנה את הפקודה של הלולאה - loop mainGameLoop

לקפיצה לתווית של mainGameLoop

jmp mainGameLoop