המשחק

# תנאים לולאות ופעולות

## 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 המשחק 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.

## 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 [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.

נגדיר במקטע הנתונים:

direction db 'r'

**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  שימו לב:  אם נצייר את הדמות במקום אליו הגענו נעמוד על גבול המשחק ה"קיר". לכן נזיז שוב את הסמן והפעם לכיוון החדש כלומר נזמן את setCursorePosition | ; change character direction from '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 |

נוסיף לפעולה הבודקת אם נגענו ב"תו" (קצה משונן) בדיקה אם נגענו בתו "|" קיר

אם כן נזמן פעולה המשנה את כיוון התנועה של השחקן.

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

באסמבלי מותר לזמן פעולה מתוך פעולה

**נזמן את הפעולה הקובעת את כיוון התנועה של השחקן מהפעולה הבודקת את התווים.**

## 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 בדיקה 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

**תנועת 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]

כעת כשחישובי התזוזה מט יותא מורכבי נרכז אותם בפעולה אחת.

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]

אם לא הקשנו על רווח נגדיל את [y\_coord]

## פעולה המחשבת את תנועת 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

**הלולאה הראשית של המשחק**:

; increase x\_cord by 1

inc [x\_cord]

cmp [chr], ' '

je jmpCar

add [y\_cord], 2

mov [color], 0Eh

jmp continue

jmpCar:

sub [y\_cord], 2

continue:

מאחר והכנסנו את חישוב הקואוקדינטות לפעולה calculateCords הבודקת מהו הכיוון של תנועת השחקןת עלינו למחוק את הפקודות הללו מהתכנית הראשית ונחליף אותן בקריאה ל calculateCords:

mov [chr], 0

**mainGameLoop:**

; delete character

mov [color], 0

call drawCharacter

; set cursor location and direction

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

**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