



פרויקט גמר בארגון המחשב ושפת סף - הערכה
חלופית - מדעי המחשב



מגישה: הילה בלומן
כיתה: י"ג
מורה: צביקה שטרקמן

תוכן עניינים:

עמוד 0 : שער.

עמוד 1 : תוכן עניינים.

עמוד 2 : הסבר על מה זה Snake

עמוד 3 : למה בחרתי בזה.

עמוד 4 : תרשים זרימה כללי של הקוד.

עמוד 5-8: הסבר על 3 פרוצדורות מרכזיות.

עמוד 9-10 : הנחיות למפעיל.

עמוד 11 : קשיים, בעיות ופתרונות.

עמוד 12 : רפלקציה וסיכום.


מה זה Snake?


תיאור כללי:

המשחק SNAKE הוא משחק לשחקן אחד, בו השחקן שולט בדמות לבנה על המסך שנראית כמו נחש. המטרה של המשחק היא להגיע לניקוד הגבוה ביותר, כל אכילת תפוח מזכה בנקודה אחת. בנוסף, כאשר הנחש אוכל תפוח הוא מתארך. אם הנחש יפגע בקיר או בעצמו הוא נפסל והמשחק נגמר.



דקה

לאחר 

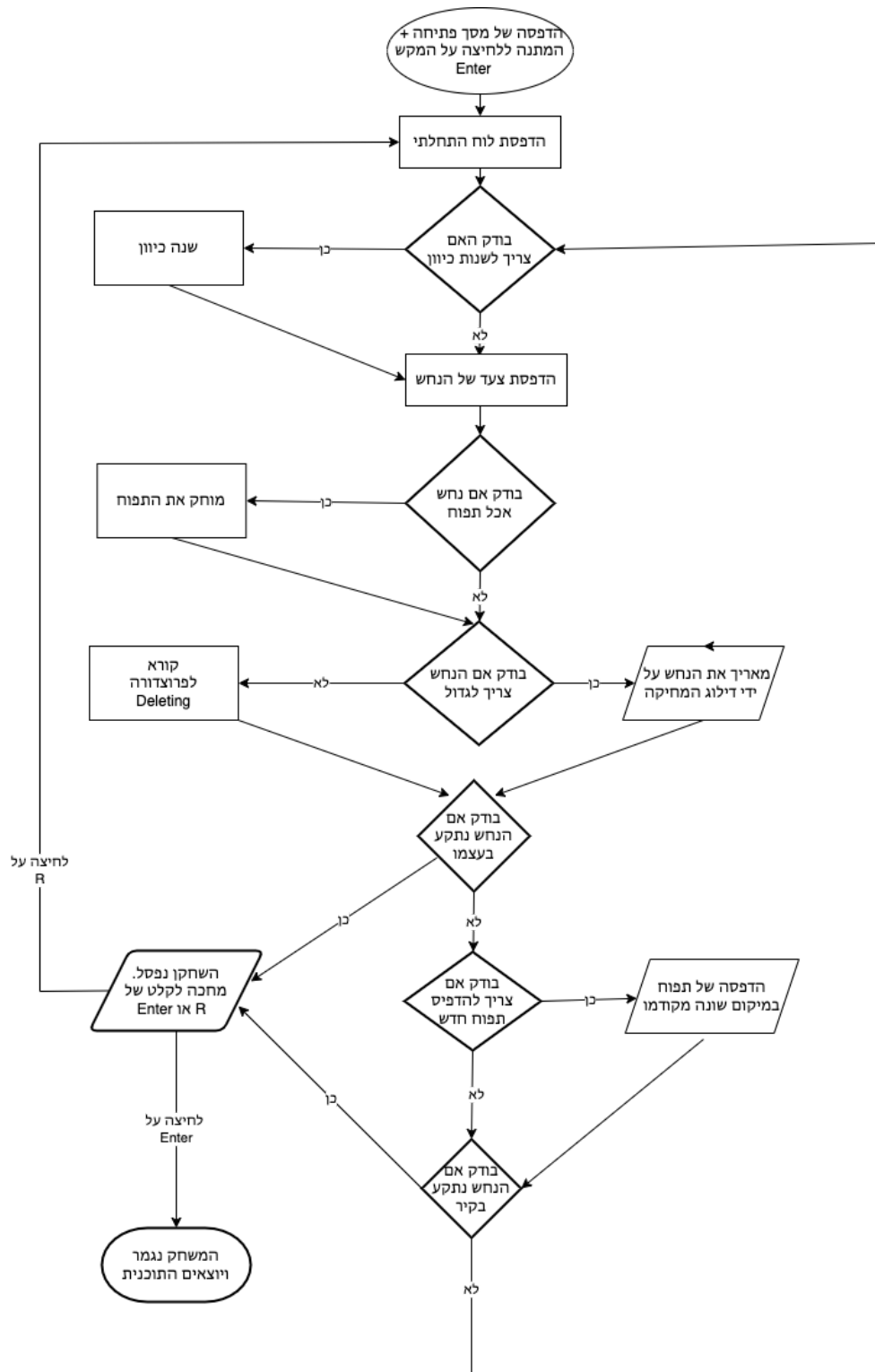
בערך של משחק 



למה בחרתי בנושא הזה?

בחרתי בנושא זה מכיוון שרצית לשחזר בקוד את אחד מהמשחקים שהייתי משחקת בהם כילדה קטנה. המשחק snake מזכיר לי את התקופות הטובות שהיו לי עם בני דודים שלי, לכן החלטתי ללכת על זה וליצור את המשחק.

תרשים זרימה (יישום הפרויקט)



פרוצדורות עיקריות

Direction

מטרתה הפרוצדורה Direction היא לקבוע את כיוון ההתקדמות של הנחש. בהתחלה נבדקת תקינות הקלט מהמשתמש (כלומר אם כיוון ההתקדמות כרגע הוא למעלה הפרוצדורה תאפשר לקבל רק קלט של ימינה או שמאלה ותתעלם משאר הקלטים). לאחר מכן הפרוצדורה משנה את ערכי המשתנים בהתאם לכיוון ההתקדמות הקיים ולקלט שקיבלה מהמשתמש במידה שהתקבל. הפרוצדורה משתמשת במשתנים שונים ומערכים:

משתנים רגילים: snakeDir, currentX, currentY, click
מערכים: delCounter, deleteY, deleteX, turns

```
proc Direction
    push ax
    push bx
    mov di, [click] ; click represent the place in the array
    cmp [turns + di - 02], 'U'
    je RL
    cmp [turns + di - 02], 'D'
    je RL
    jmp UD
RL:
    ;Check if arrow keys left or right were pressed
    cmp al, 0CDh ;Arrow right
    je move_right
    cmp al, 0CBh ;Arrow left
    jne long_jump
    jmp move_left
```

בודק מה הוא כיוון ההתקדמות הנוכחי של הנחש ולפי זה קופץ לבדיקה נכונה של הקלט.

בדיקה של הקלט במצב בוא הנחש זז למעלה או למטה

```
UD:
    ;Check if arrow keys up or down were pressed
    cmp al, 0C8h ;Arrow up
    je move_up
    cmp al, 0D0h ;Arrow Down
    je down
    jmp outp
down:
    jmp move_down
```

בדיקה של הקלט במצב בוא הנחש זז ימינה או שמאלה

דוגמה לשינוי ערכים:

```
move_left:
    mov [turns + di], 'L'
    mov [snakeDir], 'L'

    cmp [turns + di - 02], 'U' ;Check which direction the snake is currently moving
    jne DL1
    sub [currentX], 01          ;;Calculate the new position of currentX & currentY
    add [currentY], 01

    push [currentX]
    pop [deleteX + di]         ;Calculate the position and save it to deleteX[click]
    add [deleteX + di], 04
    push [currentY]
    pop [deleteY + di]         ;Calculate the position and save it to deleteY[click]
    add [deleteY + di], 03

    ;Calculate the number we need to delete before we change direction and save it to delcounter[click-01]
    push [deleteY + di]
    pop [delCounter + di - 02]
    mov ax, [deleteY + di - 02]
    sub [delCounter + di - 02], ax
    sub [deleteY + di], 03

    jmp change
DL1:
    sub [currentX], 01          ;Calculate the new position of currentX & currentY
    sub [currentY], 04

    push [currentX]
    pop [deleteX + di]         ;Calculate the position and save it to deleteX[click]
    add [deleteX + di], 04
    push [currentY]
    pop [deleteY + di]         ;Calculate the position and save it to deleteY[click]

    ;Calculate the number we need to delete before we change direction and save it to delcounter[click-01]
    push [deleteY + di]
    pop [delCounter + di - 02]
    mov ax, [deleteY + di - 02]
    sub [delCounter + di - 02], ax

    jmp change
```

לאחר כל שינוי כיוון חוקי וביצוע השינויים הדרושים (כמו בדוגמה למעלה) הקוד הבאה רץ:

```
change:
    cmp [delCounter + di - 02], 0
    jge add_click
    neg [delCounter + di - 02]

add_click:
    add [click], 02

outp:
    pop bx
    pop ax
```

בודק אם החישוב של ה- delcounter שלילי אם כן הופך אותו לחיובי (הסבר על המערך delcounter בתיאור הפונקציה Deleting)

מקדם את click אם הקלט שנקלט היה חוקי (הסבר על המשתנה בתיאור הפונקציה Deleting)

Move

לאחר שהפּרוּצדורה Direction קבעה את כיוון ההתקדמות של הנחש. הפּרוּצדורה Move תקדם את הנחש לכיוון הנכון, בתחילתה מתבצעת בדיקה של הכיוון הנוכחי של הנחש (snakeDir) וקריאה לפּרוּצדורה המתאימה. בתחילת הפּרוּצדורה נכנסים למחסנית הערכים currentX ו-currentY הערכים האלה משומשים בפּרוּצדורת שנקראות בהמשך הפּרוּצדורה (moveL, moveR, moveU, moveD)

```
proc Move
    push [currentY]
    push [currentX]
    cmp [snakeDir], 'R' ;Checking which way does the snake need to progress to
    je MR
    cmp [snakeDir], 'U'
    je MU
    cmp [snakeDir], 'D'
    je MD
;ML ;Calling the right procedure according to cmp above
    call moveL
    jmp ate_apple
MD:
    call moveD
    jmp ate_apple
MU:
    call moveU
    jmp ate_apple
MR:
    call moveR

ret
endp Move
```


Deleting

מטרת הפרוצדורה Deleting היא למחוק את הזנב של הנחש, הפרוצדורה עושה זאת בעזרת כמה משתנים:

- currentDelete שומר את המיקום במערכים לפיו יש למחוק,
- המערכים deleteX ו- deleteY שומרים את המיקום המדויק של הפיקסלים על הנחש אותם יש למחוק.
- המערך delCounter שבו שמורים כמות הפעמים שיש למחוק מאותו כיוון עד שיש שינוי בכיוון המחיקה.

```
proc Deleting
    mov si, [currentDelete]      ;move to si the place in the arrays to delete by
    mov [color], 0
;incN
    cmp [delCounter + si], 0      ;Checks if we need to move on to delete the next dircetion
    jne con
    add [currentDelete], 02
    add si, 02
con:
    cmp si, 0      ;check if this is the first dircetion
    jne turn
    call deleteRight
    jmp dec_or_not
turn:
    cmp [turns + si], 'R'      ;Checks in which dircetion we need to delete
    je BR
    cmp [turns + si], 'U'
    je BU
    cmp [turns + si], 'D'
    je BD

;BL
    call deleteLeft
    jmp dec_or_not
BD:
    call deleteDown
    jmp dec_or_not
BU:
    call deleteUp
    jmp dec_or_not
BR:
    call deleteRight

dec_or_not:
    cmp [delCounter + si], -1      ;Check if we need decrease the counter for the current turn
    je out_of_Deleting
    dec [delCounter + si]
out_of_Deleting:
    ret
endp Deleting
```

הנחיות למפעיל

איך מפעילים:

Windows - ל

צריך לפתוח את ה - Dosbox ולאחר מכן להכניס את הפקודות הבאות:

- mount c: c:/
- c:
- cd tasm/bin
- cycles = max
- tasm /zi snake
- tlink /v snake
- snake

macOS - ל

צריך לפתוח את ה - Dosbox ולאחר מכן להכניס את הפקודות הבאות:

- mount c ~/tasm/bin
- c:
- cycles = max
- tasm /zi snake
- tlink /v snake
- snake

```
Z:\>mount c ~/tasm/bin
Drive C is mounted as local directory /Users/hila/tasm/bin/

Z:\>c:

C:\>cycles = max

C:\>tasm /zi snake
Turbo Assembler Version 4.1 Copyright (c) 1988, 1996 Borland International

Assembling file:   snake.ASM
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  446k

C:\>tlink /v snake
Turbo Link Version 7.1.30.1. Copyright (c) 1987, 1996 Borland International

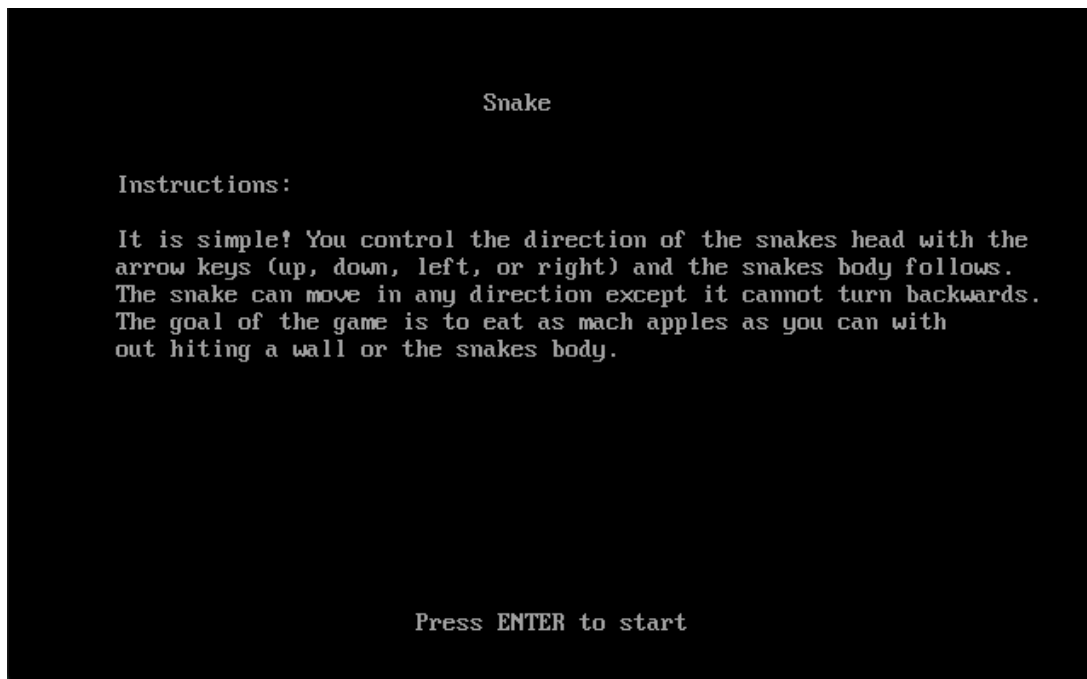
C:\>snake
```

מה צריך לקרות:

לאחר ביצוע הוראות ההפעלה למעלה, יופיע מסך הפתיחה עם הוראות המשחק, מטרת המשחק וחוקי המשחק. לאחר מכן פשוט לוחצים על Enter ומתחילים לשחק.

הוראות המשחק:

אתה שולט בכיוונו של ראש הנחש עם מקשי החצים (למעלה, למטה, שמאלה או ימינה), וגוף הנחשים עוקב. הנחש יכול לנוע לכל כיוון, אך אינו יכול להסתובב לאחור(כלומר עם הזימה הוא לא יזוז שמאלה גם אם נלחץ על המקש שמאלה). בנוסף אם הנחש מתקדם ימינה אין צורך להמשיך ללחוץ על המקש ימינה הוא מתקדם אוטומטית אם אין שינוי, לחיצה על המקש ימינה לא תשנה כלום. מטרת המשחק היא לאכול כמה שיותר תפוחים בלי לפגוע בקיר או בגוף הנחש.



מגבלות ידועות:

1. קיימת מגבלה של מקסימום 300 פניות למשחק.
2. יש מגבלה של ניקוד - הניקוד יכול להגיע רק עד 99.

קשיים, בעיות ופתרונות

לאורך הפרויקט לא היו לי הרבה בעיות וקשיים, נתקלתי בשני אתגרים מעניינים.

מחיקת הנחש לאחר יותר מפניה אחת -

הקושי היה שהגרסה הראשונית של המחיקה לא אפשרה יותר מפניה אחת בכל פעם. הגרסה לא אפשרה את זה כיוון שהיא הסתמכה על `currentX`, `currentY` ועל כמות קבועה של פעמים שיש לחזור על המחיקה של הזנב.

הפתרון שלי לבעיה זו היה להשתמש במערכים אשר ישמרו בכל פנייה את הנתונים הנדרשים (מיקום הפיקסל שממנו מוחקים, כמות המחיקות שיש לבצע, וכיוון המחיקה) ושני משתנים, אחד אשר שומר את המיקום במערכים לפיו יש למחוק והשני ששומר המיקום במערך שבו הערכים לאחר פניה יוכנסו. בעזרת המערכים ושני המשתנים הצלחתי לקודד קוד למחיקה של זנב הנחש שעובד גם במצב של פניות מרובות לא משנה מהו אורכו של הנחש.

הדפסה רנדומלית של תפוחים - להדפיס תפוחים במקומות שונים על המסך (רנדומליים), קושי זה נוצר משתי סיבות:

1. כתיבת קוד לקבלת מספרים רנדומליים - ניסיתי לכתוב קוד שעושה זאת והתקשיתי מאוד.
2. לדאוג שהתפוח לא יודפס מחוץ לקירות או עליהם - השימוש בקוד שבודק את זה היה מאט מאוד את התוכנית ולא יעיל.

פתרון - בסוף השתמשתי במערך של מיקומי תפוח שונים מוכנים מראש ובקוד שיוצר מספרים רנדומלי (שמצאתי בספר). בעזרת המספרים הרנדומלים נבחר כל פעם מיקום אחר במערך שלפיו מודפס תפוח.

רפלקציה וסיכום

איך היה לי הפרוייקט:

התחלתי את הפרוייקט עם הרבה ביטחון בידע שלי וההבנה שלי החומר, אבל לאורך הזמן הבנתי שחסר לי ידע כדי לקודד את הפרוייקט הזה, למרות זאת המשכתי לעבוד, לטעות, לתקן וללמוד. נתקלתי בכמה בעיות בתכנון ובבאגים בקוד בעיקר על המיקום של הנחש (התקדמותו ומחיקתו), את רובם פתרתי תוך כמה דקות או שעות, אך היו מספר בעיות שלקחו לי כמה ימים לפתור רעיונית ולממש. ואפילו במסגרת הפרוייקט תכננתי אלגוריתם לפתרון הבעיה.

לאורך הפרוייקט היו לי עליות וירידות רבות, זמן רב הושקע בפרוייקט ואני שמחה מאוד מהתוצר הסופי ומתגאה בעובדה שקודדתי משחק בעצמי.

מה הייתי עושה אחרת:

בתחילת הפרוייקט קפצתי לתוך הפרוייקט ישר התחלתי לכתוב את הקוד לפי הרעיון שהיה לי. אם הייתי צריכה לעשות את הפרוייקט מחדש לא הייתי מתחילה בכתיבת הקוד לאור התכנון הראשוני, אלה הייתי כותבת תכנון יותר מסודר על דף, רושמת איך הייתי עושה כל חלק בנפרד. אם הייתי עושה את זה בדרך הזו הייתי חוסכת לעצמי הרבה זמן וכאבי ראש.

מטלת המשך

המטלה שקיבלתי מצביקה לאחר הצגת הפרויקט היה להפוך את הנחש למהיר יותר. עשיתי זאת על ידי הדפסה ומחיקה של שתי עמודות/שורות של פיקסלים במקום אחת. בעקבות שינוי זה הייתי צריכה לשנות עוד כמה דברים בקוד על מנת שהוא יפעל כמו שצריך.