# פקודות בקרה – ביצוע חוזר

## לולאות

לעיתים קרובות אנחנו צריכים לבצע פעולה כלשהי מספר מוגדר של פעמים.

כדי לבצע פעולה מספר פעמים:

1. יש צורך לשמור ברגיסטר כלשהו את כמות הפעמים לביצוע (רגיסטר או משתנה)
2. ליצור מונה שימנה את מספר הפעמים שביצענו את הפעולה (נשתמש ברגיסטר cx)
3. להשוות את כמות הפעמים שבוצעה הפעולה אם מספר הפעמים לביצוע
4. אם מתקיים שוויון לצאת מהלולאה
5. ואם לא מתקיים שוויון – לקפוץ אחורה לפקודות לביצוע חוזר.

ביצוע פעולה מוגדרת מספר פעמים – נקראת לולאה (Loop), כיוון שפעולה זו נפוצה למדי, ישנה פקודה מיוחדת שעושה בשבילנו חלק מהעבודה.

פקודת loop מבצעת את הפעולות הבאות:

- מפחיתה 1 מערכו של cx.

- משווה את cx לאפס.

אם אין שוויון, כלומר, ערכו של cx אינו אפס, מבצעת jmp ל-label שהגדרנו.

**loop SomeLabel**

זהה להוראות הבאות:

dec cx

cmp cx, 0

jne SomeLabel

פקודת loop מבצעת את הפעולות הבאות- לפי הסדר הבא:

* מפחיתה 1 מערכו של cx
* משווה את cx לאפס
* אם אין שוויון (כלומר, ערכו של cx אינו אפס) - מבצעת jmp ל-label שהגדרנו

קטעי הקוד הבאים זהים

|  |  |
| --- | --- |
| LableName:  dec cx ;הפחתה ב - 1  cmp cx, 0 ;האם האינדקס שווה ל - 0  jne LableName ;חזרה לתחילת הלולאה | SomeLabel:  ; פקודות לביצוע  loop SomeLabel |

|  |  |
| --- | --- |
| **פקודת הדפסה של 'x' למסך בעזרת תווית** | **פקודות הדפסה של 'x' למסך בעזרת לולאה.** |
| mov cl, 0  printX: ; print ‘x’ to the screen  mov dl, 'x'  mov ah, 2h  int 21h  **inc cl**  **;comp if cl equal to 10**  **cmp cl, 10**  **jne printX** **; junp if not eual** | mov cx, 10  PrintX: ; print ‘x’ to the screen  mov dl, 'x‘  mov ah, 2h  int 21h  loop PrintX |
|  | הלולאה מבצעת אוטומטית את הפעולות הבאות:   * מפחיתה 1 מערכו של cx * משווה את cx לאפס * אם אין שוויון (כלומר, ערכו של cx אינו אפס) - מבצעת jmp ל-label שהגדרנו. |

כמה פעמים יודפס 'x' למסך?

mov [TimesToPrintX], 0

mov cx, [TimesToPrintX]

PrintX:

mov dl, 'x'

mov ah, 2h

int 21h

loop PrintX

התשובה: 65,536 פעמים

פקודת ה-loop קודם מורידה את ערכו של cx ורק אחר כך בודקת אם הוא גדול או שווה לאפס!

מאחר והלולאה מפחיתה את המונה cx אם הוא מאופס (cx=0)

הפחתה של 1 משני בתים 0000 0000 0000 0000 ⇦ 65,535 (הייצוג של -1 [256\*256-1])

לאחר ההפחתה הבאה הייצוג של cx ⇦ 65,534 וכך הלאה עד שערכו של cx יתאפס.

216 ⇨ 65,536 [1 0000 0000 0000 0000 -1 ⇨ 1111 1111 1111 1111]

איך נתקן את התכנית?

mov [TimesToPrintX], 0

xor cx, cx

mov cx, [TimesToPrintX]

**cmp cx, 0**

**je ExitLoop**

PrintX:

mov dl, 'x'

mov ah, 2h

int 21h

loop PrintX

**ExitLoop:**

**…**

לפני הכניסה ללולאה נבדוק את ערכו של cx אם הוא שווה לאפס לא ניכנס ללולאה.

# סדרת פיבונצ'

סידרת פיבונצ'י: סדרת פיבונצ'י מוגדרת באופן הבא- האיבר הראשון הוא 0, האיבר השני הוא 1, כל איבר הוא סכום שני האיברים שקדמו לו.

צרו תכנית שמחשבת את חמשת המספרים הראשונים בסדרת פיבונצ'י ושומרת אותם במערך בעל 5 בתים. בסיום התוכנית המערך צריך להכיל את הערכים הבאים:

0,1,1,2,3,

IDEAL

MODEL small

STACK 100h

יצרנו מערך מאופס של 5 בתים

איפסנו את רגיסטר ax בו נשתמש לפעולות החיבור של איברי הסדרה.   
והעברנו לרגיסטר al את הערך 1.

העברנו לרגיסטר cx את הערך 4 (בסדרה 5 איברים, אבל הראשון הוא 0 ונבצע רק 4 פעולות חיבור).

העברנו לרגיסטר bx את הכתובת של תחילת המערך.

DATASEG

fibArray db 5 dup (0)

CODESEG

start:

mov ax, @data

mov ds, ax

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

xor ax, ax

mov cx, 4 ; loop index

mov al, 1 ; number to add

mov bx, offset fibArray

NumToArray:

**inc bx**

**mov [bx], al**

**add al, [bx-1]**

לולאת חיבור האיברים:

הגדלנו את bx ב – 1 כדי לעבור לתא הבא.

רגיסטר al שומר את הערך של התא שרגיסטר bx מצביע עליו.

הוספנו לרגיסטר al את הערך שבתא הקודם לו במערך.

al = al + [bx-1]

al = arry[1] + arry[0]

al = arry[2] + arry[1]

al = arry[3] + arry[2]

al = arry[4] + arry[3]

סיום לולאה:

(בדיקת תנאי והוספת ל – cx1)

loop NumToArray

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

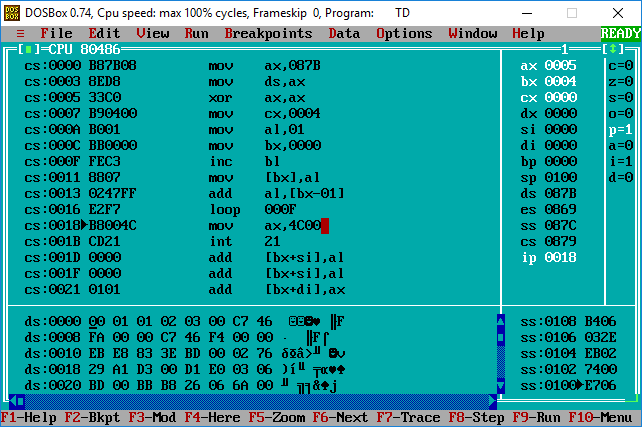
exit:

mov ax, 4c00h

int 21h

END start

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 4 | 3 | 2 | 1 | 0 | ax - al | bx | cx | loop |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 4 |  |
|  |  |  | 1 | 0 | 1+0 = 1 | 1 |  | 1 |
|  |  | 1 |  |  | 1+1 = 2 | 2 |  | 2 |
|  | 2 |  |  |  | 2+1 =3 | 3 |  | 3 |
| 3 |  |  |  |  | 3+2=5 | 4 |  | 4 |



# פקודות בקרה – ביצוע חוזר

## לולאות

## תרגילים bg6 + bg7

לעיתים קרובות אנחנו צריכים לבצע פעולה כלשהי מספר מוגדר של פעמים.

כדי לבצע פעולה מספר פעמים:

1. יש צורך לשמור ברגיסטר כלשהו את כמות הפעמים לביצוע (רגיסטר או משתנה)
2. ליצור מונה שימנה את מספר הפעמים שביצענו את הפעולה (נשתמש ברגיסטר cx)
3. להשוות את כמות הפעמים שבוצעה הפעולה אם מספר הפעמים לביצוע
4. אם מתקיים שוויון לצאת מהלולאה
5. ואם לא מתקיים שוויון – לקפוץ אחורה לפקודות לביצוע חוזר.

ביצוע פעולה מוגדרת מספר פעמים – נקראת לולאה (Loop), כיוון שפעולה זו נפוצה למדי

ישנה פקודה מיוחדת שעושה בשבילנו חלק מהעבודה.

פקודת loop מבצעת את הפעולות הבאות:

- מפחיתה 1 מערכו של cx.

- משווה את cx לאפס.

אם אין שוויון, כלומר, ערכו של cx אינו אפס, מבצעת jmp ל-label שהגדרנו.

**loop SomeLabel**

נקצר את התכנית שלנו כך שהדמות תזוז בעזרת ללולאה.

נוסיף משתנה שישמור את הערך של cx – מונה הלולאה:

בסגמנט הנתונים נוסף הצהרה על משתנה:

cx\_keeper dw 10 ; variable to keep cx in loop

בתכנית שלנו כמה שלבים:

1. שרטוט מסך הפתיחה  
   המתנה למקש  
   ניקוי המסך ע"י מעבר לתצוגה גרפית
2. שרטוט לוח המשחק  
   המתנה למקש
3. ציור הדמות הראשונה למסך (מיקום הסמן, ציור הדמות במיקום הסמן)  
   המתנה למקש.
4. מחיקת הדמות (ע"י ציור הדמות בשחור על הצבעונית)  
   הזזת הסמן למיקום החדש (שינוי המיקום על ציר x או על ציר y או בשניהם)  
   ציור הדמות בצבע במיקום הסמן.
5. חזרנו על סעיף 4 כמה פעמים כדי שהדמות שלנו תוזז כמה פעמים על המסך.

נקצר את התכנית ע"י שימוש בלולאה.

## יצירת לולאה

נוסיף משתנה שישמור לנו את מונה הלולאה

בתוך הספיקות נעשה שימוש ברגיסטר cx שהוא גם מונה הלולאה,

נשמור את מונה הלולאה למשתנה

ונשלוף אותו לקראת סוף הלולאה.

במקטע הנתונים נוסיף משתנה:

cx\_keeper dw 10 ; variable to keep cx in loop

נשמור רק את הפסיקות של תזוזת הדמות שלנו באלכסון,   
גם על ציר ה – x וגם על ציר ה – y.

נמחק את כל הפעמים האחרות בהם כתבנו פסיקות לתזוזות הדמות.

נוסיף לולאה שתעטוף את כל הפקודות.

בתחילה תווית של - mainGameLoop

ובסוף הפקודה של Loop MainGameLoop

במקטע הקוד: לאחר שרטוט הדמות בפעם הראשונה ובדיקה האם הקשנו על מקש 'q', נוסף את הפקודות:

שימו לב 🎔, חלק מהפקודות קיימות

**mov cx, 10 ; loop count**

נקבע את מונה הלולאה ל –10

והתווית לתחילת הלולאה

**mainGameLoop:**

בפסיקה יש שימוש ברגיסטר cx לכן **נשמור** את הערך שלו למשתנה

**mov [cx\_keeper], cx**

; delete character & move character by 1 step

; draw blank – character on cursor position

שינוי הצבע של השחקן לשחור – ומחיקת הדמות

mov [color], 0

mov ah, 9

mov al, 0 ; al = character to display

mov bx, [color] ; bh = Background bl = Foreground

mov cx, 1 ; cx = number of times to write character

int 10h

; increase x\_cord by 1 and y\_cord by 2

inc [x\_cord]

שינוי הערכים של מיקום הסמן על ציר x, על ציר y ושינוי הצבע

add [y\_cord], 2

mov [color], 0Eh

; print chararcter

; set cursore location

mov dh, [y\_cord] ; row

הזזת הסמן למיקום החדש

mov dl, [x\_cord] ; column

mov bh, 0 ; page number

mov ah, 2

int 10h

ציור הדמות במיקום החדש

; draw smiley - ascii 2 at cursor position

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

; waits for character

mov ah, 0h

המתנה למקש

int 16h

; check if user asks to quit

האם המשחק רוצה לצאת?

cmp al, 'q'

je end\_game

**mov cx, [cx\_keeper]**

נחזיר את הערך של מונה הלולאה לרגיסטר cx

והפקודה של סוף איטרציה של הלולאה.

**loop mainGameLoop**

end\_game:

; text mode

mov ax, 2h

int 10h

הריצו את התכנית ובדקו שהדמות שלכם זזה 10 פעמים באלכסון.

בדקו האם בהקשה על q אתם יוצאים לפני סוף הלולאה.

ניתן לשנות את הלולאה, ללולאה של משתנה ללא רגיסטר cx.

שמרו את הקובץ בשם חדש והמשיכו לעבוד על הקובץ החדש.

# לולאה מותנית

לולאה זו מבוססת על משתנה אשר משמש מונה הלולאה,

מגדילים (או מקטינים) אותו בכל איטרציה של הלולאה,

משווים אותו לערך הרצוי ובהתאם לתוצאה

קופצים לתווית של תחילת הלולאה או ממשיכים הלאה.

ניצור משתנה בשם loop\_num שישמש "מונה" הלולאה

נגדיל אות איטרציה ב – 1 ונשווה אותו לערך 10

נוסף את המשתנה למקטע הנתונים

loop\_num db 0

במקטע הקוד

לפני התווית של הלולאה, נמחק את הפקודה המשימה את הערך 10 לרגיסטר cx

מחק את ההוראה

mov cx, 10 ; loop count

בתוך הלולאה נמחק את השמירה של רגיסטר cx ל – משתנה cx\_keeper

mov [cx\_keeper], cx

נשאיר את התווית של הלולאה, ונרד לסופה.

נמחק את הפקודה המציינת שזו לולאה.

loop mainGameLoop

בסוף הלולאה נעדכן את המשתנה loop\_num

נבדוק האם הוא שווה לערך 10

אם לא ⇦ נחזור לתווית של mainGameLoop

מבנה סוף הלולאה:

inc[loop\_num]

cmp [loop\_num], 10

jb mainGameLoop

שנו את הלולאה כך שהשחקן יזוז 5 צעדים למעלה וצעד אחד ימינה בכל הקשה.

**mainGameLoop:**

; move charcter by 1 step

; delete chraracter

; draw blank - ascii 0 at cursor position

mov [color], 0

mov ah, 9

mov al, 0 ; al = character to display

mov bx, [color] ; bh = Background bl = Foreground

mov cx, 1 ; cx = number of times to write character

int 10h

; increase x\_coord by 1 and y\_cord by 2

; inc [x\_cord]

; add[y\_cord], 2

dec [y\_cord]

mov [color], 0Eh ; change in to color

; print chararcter

; set cursore location

mov dh, [y\_cord] ; row

mov dl, [x\_cord] ; column

mov bh, 0 ; page number

mov ah, 2

int 10h

; draw smiley - ascii 2 at cursor position

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

; waits for character

mov ah, 0h

int 16h

; check if user asks to quit

cmp al, 'q'

je end\_game

**inc[loop\_num]**

; cmp [loop\_num], 10

**cmp [loop\_num], 5**

**jb mainGameLoop**

end\_game:

; text mode

mov ax, 2h

int 10h